Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería Informática

Proyecto Fin de Carrera

ChessAsin, servidor de ajedrez por correspondencia

Autor: Gil Izquierdo, Enrique Juan

Tutor: Linares López, Carlos

Tabla de Contenidos

Índice de Tablas

Índice de Figuras

Agradecimientos:

A mi madre, por su (im)paciencia e insistencia.

A mi familia y amigos, por no dejar de preguntar sobre el proyecto.

A Vidal, Álvaro, Wikipedia e Internet por lo que me han enseñado sobre ajedrez.

A Rafa, por la idea y la constante educación.

Y a Carlos, por darme una segunda oportunidad.

This document is Copyright ©2011, Enrique J. Gil Izquierdo.

Capítulo 1: Introducción

El ajedrez es uno de los juegos más antiguos, con más de dos mil años de historia, y uno de los más practicados en la actualidad. Sin importar el paso del tiempo, este deporte continúa fascinando a las nuevas generaciones, sobreviviendo a la aparición de nuevas formas de entretenimiento. Tal es su importancia que, aparte de ser deporte nacional en gran cantidad de países, cuenta con múltiples torneos internacionales, siendo los jugadores más importantes mundialmente conocidos [BRIT 1].

A pesar de su larga historia, el ajedrez ha continuado evolucionando, con adaptaciones de las reglas o los sistemas de clasificación y campeonato. Pero ninguno de los cambios ocurridos en toda su historia puede compararse con el que ha sufrido en los últimos cuarenta años. Prácticamente desde los inicios de la informática se han dedicado buena parte de sus recursos a analizar el ajedrez, como interesante problema matemático. Hoy día casi cualquier máquina puede derrotar a maestros internacionales, se juegan millones de partidas por Internet al día y se estudian partidas o problemas a través de interfaces web.

Y en los últimos años la explosión ha crecido, con la profusión de plataformas móviles con gran capacidad de procesamiento. Ahora no es necesario permanecer sentado frente a un ordenador para jugar una partida, sino que se puede hacer en cualquier lugar, usando teléfonos móviles, ordenadores ultraportables o tabletas. Cada una de estas plataformas, además, ofrece sistemas de distribución con pocas barreras a la hora de vender aplicaciones para ellas. Esto genera una gran competencia entre desarrolladores, creando un entorno dinámico que favorece a aquéllos que mantienen el equilibrio entre la facilidad de desarrollo y la velocidad de adaptación a las nuevas posibilidades.

Entre las principales características que se pueden encontrar en las aplicaciones de ajedrez más populares podemos encontrar el juego -individual, en red o asíncrono-, el estudio de partidas y la resolución de problemas. En lo que respecta al juego en red, cabe destacar que la mayor parte de servidores utilizados fueron desarrollados en los 90, programados en lenguajes como C/C++ o en plataformas web ya anticuadas. Esto implica que estos servidores son más difíciles de adaptar y modificar a la hora de añadir nueva funcionalidad, requieren baterías de pruebas mucho más complejas y, en general, son menos agradables para los programadores.

En el caso particular de las plataformas móviles esto crea un problema, pues éstas suelen ser programadas en lenguages y entornos de alto nivel, como Cocoa, Java o .NET. Así, los

desarrolladores interesados en trabajar en estos entornos se ven obligados a mantener unos servidores creados con tecnologías antiguas, a la vez que trabajan en clientes mucho más avanzados.

ChessAsin nació con la intención de crear una nueva plataforma abierta para los jugadores de ajedrez, desarrollada en un lenguaje de programación moderno y aprovechando tecnologías de comunicación en red más potentes. Esto simplificará la creación de clientes, dadas las facilidades que ofrece para aprovechar las características de sistemas operativos como Android o Windows Phone 7. Además, el uso de lenguajes de alto nivel permitirá introducir mejoras en el servidor a un ritmo mayor, asegurando una mayor competitividad en un entorno tan hostil como el actual.

Una vez terminado, el servidor se liberará como proyecto de código libre bajo la licencia GPLv3. La dirección del proyecto es: http://sourceforge.net/projects/chessasin/.

Capítulo 2: Estado de la cuestión

Este capítulo presenta el juego de Ajedrez, presuponiendo que el lector conoce sus reglas básicas, para más tarde analizar la relación de la informática y, específicamente, las plataformas móviles modernas con el juego. Finalizaremos con un análisis de las plataformas móviles más populares, sus medios de distribución de aplicaciones y entornos de desarrollo.

En caso de desconocer las reglas del ajedrez, puede encontrarlas en [FIDE 2 Laws].

2.1 Introducción al ajedrez

El ajedrez es uno de los juegos más antiguos y populares del mundo, conocido en múltiples culturas desde hace cientos de años. Desde sus primeras versiones hasta la actualidad, reglas, piezas y tablero han ido adaptándose a cada cultura, pero siempre han mantenido una base reconocible, así como una gran facilidad para mantener y crear adeptos. Su atractivo se debe en gran medida al equilibrio entre la simplicidad de sus reglas y la complejidad de las partidas que pueden surgir de ellas.

En términos básicos, es un juego para dos jugadores que transcurre en un tablero cuadrado de 8 por 8 casillas decoradas con dos colores distribuidos alternativamente. Los colores de las casillas, al igual que los jugadores y sus piezas, son identificados como blanco y negro, aunque sea común encontrar versiones del juego con esquemas de colores diferentes. Un tablero de ajedrez y las piezas pueden estar hechos de diversos materiales, siendo los más comunes madera, metal o piedra.

Desde su versión original, el ajedrez ha servido como representación simplificada del conflicto que se da en un campo de batalla o una guerra, con cada pieza representando a una unidad de combate. Como en las antiguas guerras, el objetivo final es obligar al rey enemigo a rendirse, consiguiendo ponerle en una posición de la que no pueda escapar. El rey es la única pieza del tablero que no es posible capturar, siendo cualquier movimiento que le deje bajo amenaza ilegal. A pesar de su gran importancia, es junto con el peón la pieza más limitada en su movimiento, dependiendo del resto del ejército para defenderse [BRIT 1 History].

Cabe destacar que, tras siglos de adaptaciones y modificaciones de las reglas, las partidas de ajedrez guardan hoy día poco parecido con las batallas que originalmente simulaban, más allá de una libre idealización.

2.1.1 Historia

Aunque durante siglos se discutieron los posibles orígenes de lo que hoy conocemos como ajedrez, se acepta como correcta la teoría de que tanto éste como el resto de juegos similares que se pueden encontrar a lo largo del mundo partieron del pasatiempo indio *chaturanga*. Creado en el siglo VI A.D., se extendió rápidamente de civilización en civilización, con o sin modificaciones o mezcla con otros juegos locales.

El *Chaturanga*, referencia poética a los cuatro componentes de un ejército[1], pretendía ser una idealización de un campo de batalla. Lo más destacable era el hecho de que el rey, a pesar de ser una pieza fundamental, se encontraba tanto o más limitada que el resto de su armada.

Como resultado de los constantes conflictos y conquistas de los siglos VI y VII, el *chaturanga* era conocido en toda Asia Menor y Sur. Más tarde entró en China, donde se cree que produjo la evolución del antiguo *Xiangqui* a su versión actual, para más tarde llegar a Japón, donde fue adoptado con notables cambios, pasando a llamarse *Shogui*.

El imperio persa también asimiló el juego, adaptándolo a un estilo más moderno y creando algunos de los términos más reconocibles del ajedrez moderno. Por ejemplo, jaque, check o échecs derivan del grito "*Sha!*", título del rey persa, proferido cuando éste era amenazado en el juego. Después de caer el Imperio Persa bajo el control de los musulmanes el ajedrez se extendió por todos sus territorios, hasta llegar a al-Ándalus, donde conservó su popularidad tras la Reconquista.

Rápidamente todos los reinos europeos cayeron bajo el encanto del juego, ya prácticamente en su forma moderna, y empezaron a surgir aficionados y clubes dedicados a estudiar sus propiedades.

Con el paso del tiempo se fueron introduciendo modificaciones sobre el comportamiento de ciertas fichas, con la reina y el alfil pasando de mover una única casilla a su actual capacidad ofensiva o los peones ganando la habilidad de avanzar el doble en su salida. Otros cambios de gran relevancia fueron la consideración de que el ahogamiento se convirtiese en empate, en lugar de derrota, o las condiciones especiales de tablas.

2.1.2 El ajedrez hoy

En la actualidad el ajedrez ha alcanzado la categoría de deporte nacional en varios países,

1 Elefantes, carros de combate, caballería e infantería. *Chaturanga* siginifica "de cuatro miembros".

especialmente en Europa del Este, donde se practica desde muy temprana edad. Millones de personas juegan a diario, ya sea como meros aficionados o como profesionales, en alguna de sus variantes.

Tal es su importancia y el deseo de los jugadores de medirse entre sí que se ha creado un organismo internacional encargado de regular los encuentros oficiales: la *Fédération Internationale des Échecs* (FIDE) (ver sección 2.1.4.1). Entre sus múltiples responsabilidades se encuentra la de establecer las reglas de juego para encuentros oficiales.

Pero, aparte de los encuentros oficiales, jugados cara a cara sobre un tablero, también se juega usando otros sistemas, ya sea el correo, ordenador, servidores web o móviles. Muchas de las partidas jugadas con estos métodos no están reguladas por la FIDE, pero aún así se basan en sus reglas o en pequeñas variaciones de las mismas. En el caso particular del ajedrez por correo, tanto normal como electrónico, es la *International Correspondence Chess Federation* (ICCF) (ver sección 2.1.4.2)la encargada de establecer las reglas a aplicar en encuentros oficiales, teniendo en cuenta las particularidades propias del medio de comunicación.

2.1.3 Modos de juego

Como se ha explicado anteriormente, existen en la actualidad múltiples formas de participar en encuentros entre ajedrecistas. A continuación explicaremos las características principales de los más importantes.

2.1.3.1 Sobre el tablero

En su formato clásico, el ajedrez se juega entre dos jugadores, cara a cara, moviendo las piezas físicamente sobre el tablero. Las reglas más importantes que se aplican exclusivamente a este modo de juego se refieren a la prohibición de tocar o mover las piezas del tablero, salvo para ejecutar el movimiento deseado.

En partidas no amistosas se utilizan mecanismos para controlar el tiempo que cada jugador dedica a pensar un movimiento o para el total de la partida. A cada jugador se le otorga un tiempo máximo en el que debe ejecutar un número mínimo de movimientos, con la posibilidad de sumar una cantidad de tiempo extra por cada movimiento realizado. Con objeto de controlar esto se han desarrollado una variedad especial de relojes, denominados *de ajedrez*, preparados para esta tarea. Tras cada jugada el jugador está obligado a detener su reloj e iniciar el de su oponente, aunque en

todos los relojes de ajedrez modernos el inicio del reloj opuesto es automático.

En los encuentros oficiales no se permite ningún tipo de consulta sobre la partida, ni mecanismos de ayuda en la toma de decisiones, como un tablero adicional en que probar jugadas.

2.1.3.2 Por correspondencia

Ésta no es más que una adaptación de las reglas básicas para permitir a dos jugadores enfrentarse a gran distancia. El cambio más importante, aparte del hecho de que no se juega en persona, es que el tiempo de partida no está limitado, pero sí el que se puede dedicar a cada movimiento. Antiguamente podía llegar a ser de un mes o más, por su dependencia con los antiguos sistemas de correo, aunque los avances tecnológicos han mejorado considerablemente esta situación. Debido a esto, las partidas pueden alargarse durante años. Otra modificación popular es eliminar la regla de tablas de los 50 movimientos, ya que el tiempo que los jugadores deseen dedicar a una partida no tiene por qué ser limitado.

Dadas estas condiciones y lo prolongados que llegan a ser los juegos, los aficionados a esta modalidad acostumbran a mantener varias partidas por correspondencia a la vez.

Uno de los principales atractivos de este modo de juego es que permite medirse contra rivales de zonas distantes, sin la necesidad de trasladarse uno de los jugadores. De hecho, los antiguos jugadores profesionales acostumbraban a jugar una serie de partidas por correspondencia con otros maestros, para probarles y decidir si un enfrentamiento cara a cara, con los costes que conllevaría, valdría la pena.

Dado que el tiempo disponible para tomar una decisión es considerable y no es posible controlar las acciones de cada jugador, es habitual que se permita recurrir a asistencias externas, como tableros o programas informáticos. Normalmente, los jugadores u organizadores del torneo acuerdan antes del comienzo si esto se permite o no.

2.1.3.3 En línea

Los dos contrincantes se enfrentan desde dispositivos electrónicos, con los movimientos de cada uno de ellos siendo comunicados al instante al otro. Dado que la cantidad de información a intercambiar es limitada, no requiere conexiones especialmente rápidas. Como mucho, cada uno de los jugadores tardará unos segundos en conocer la jugada de su oponente. Para jugar de esta forma ambos jugadores deben usar aplicaciones capaces de comunicarse entre sí y contar con la

información necesaria para conectar con el oponente. En los inicios de Internet la dirección IP sería necesaria, aunque hoy día es más común el uso de sistemas intermedios que pongan en contacto a ambos jugadores.

En este tipo de juego se suelen aplicar las normas de ajedrez normal, con límites de tiempo o no.

Existen gran cantidad de aplicaciones y páginas web que dan este servicio, tanto de pago como gratuitas, y suelen crear grandes comunidades a su alrededor.

2.1.3.4 Por correo electrónico

En la actualidad es más común que se utilice el correo electrónico, en lugar del ordinario, por comodidad y precio. Debido a la popularización de este mecanismo de juego, la ICCF extendió sus normas para regular también esta modalidad, actualizando y creando nuevas reglas para adaptarlas mejor a sus características.

También es habitual jugar por correo electrónico usando servidores intermediarios. Esto añade dos ventajas de gran importancia: privacidad y centralización. Al usar un servidor intermedio que envía los correos, los jugadores no necesitan dar a conocer la dirección de correo a aquéllos contra los que juegan, sólo el identificador de cuenta. Además el servidor puede ofrecer servicios adicionales, como revisar partidas de otros jugadores, evaluar variantes o compartir la partida con otros usuarios.

Otros servidores van más allá, dejando de lado el correo electrónico y dando acceso a funcionalidad similar a través de aplicaciones específicas. Puesto que la aplicación puede gestionar la información de la partida y mostrar el tablero, esto facilita el juego, pues éste puede ser analizado directamente, sin necesidad de *parsear* correos electrónicos.

2.1.3.5 Asíncrono

Esta modalidad de juego mezcla características del juego en tiempo real y por correspondencia, permitiendo encuentros relajados, pero más rápidos que por correo. Aunque este estilo de juego no tiene un nombre específico, hemos decidido darle el de *asíncrono* por la similitud con otros juegos que en la industria del videojuego se conocen así [ASYNC 3].

La idea básica es limitar el tiempo que cada jugador puede dedicar a pensar un movimiento a un plazo determinado, comunicando la jugada al oponente sólo al final de dicho lapso, sin importar cuándo se ejecutó realmente. Esto permite que ambos jugadores sepan exactamente cuándo van a recibir la información, evitando problemas de franjas horarias. También elimina la posibilidad de

desconcertar al contrario con movimientos especialmente rápidos, presionándole a mantenerse constantemente atento a cambios en el juego.

La partida más importante jugada en este formato ha sido la de Kaspàrov contra El Mundo, organizada por MSN en 1999. En ella Kaspàrov, entonces campeón del mundo, se enfrentó a todos los jugadores del mundo que desearan participar, aconsejados por 5 jóvenes promesas. Cada consejero recomendaría un movimiento para cada turno y se ejecutaría aquél que más votos recibiera, aunque no se encontrara entre los recomendados. Cada equipo contaba con 24 horas para decidir el movimiento a realizar, y sólo al final de ese plazo de 24 horas se comunicaba al oponente la decisión. Kaspàrov ganó tras la rendición del equipo mundial en el movimiento 62, cuatro meses después de que la partida comenzase[2].

2.1.3.6 Problemas

Los problemas de ajedrez suponen un cambio más radical al sistema de juego habitual. En primer lugar, los problemas son una prueba para un único jugador, al que se reta a alcanzar una meta determinada, normalmente un mate en determinados movimientos, moviendo el jugador ambos colores. Por tanto, el punto de partida es una configuración de tablero propia para cada problema, con pocas piezas en juego y posiciones comprometidas para blancas y negras. También existen variantes más complejas que requieren que el jugador descubra cuál fue la última jugada o, por ejemplo, si un jugador ha enrocado o no.

Usados como herramienta de estudio, los problemas suelen puntualizar el efecto de ciertas estrategias o la fuerza de piezas específicas en situaciones propicias o adversas. Puesto que la condición a cumplir más común es el jaque-mate, sirven como entrenamiento para sacar todo el partido posible de las piezas que se tengan. Este ayuda a comprender posiciones de riesgo o poder, y como explotarlas estas condiciones.

No es raro encontrar problemas basados en partidas famosas, bien recreando el final de la misma o variantes sobre situaciones que alguno de los jugadores evitó. Pero lo más común es encontrarse con problemas especialmente diseñados. Existen jugadores de ajedrez especializados en el diseño de estos problemas, algunos de ellos tan reconocidos que se les otorga el título de Gran Maestro en composición de problemas.

2 Comentario completo de la partida, por Daniel King, en http://web.archive.org/web/20020414122910/fdl.msn.com/zone/kasparov/gameanalysis.txt

Existen gran cantidad de páginas web dedicadas a la propuesta y resolución de problemas de este tipo, contándose entre las más famosas ChessTempo[3] o ChessGames[4], así como aplicaciones para todo tipo de plataformas.

2.1.3.7 Ajedrez mágico

Ésta es una variante especial, aplicable a cualquiera de las anteriormente explicadas. Consiste en la sustitución de una o más de las piezas clásicas del ajedrez por otras con habilidades especiales, como la de capturar piezas pasando sobre ellas, o movimientos compuestos, como por ejemplo una pieza que pueda mover como un caballo o un rey en cualquier momento.

Uno de los campos que más provecho sacan de estas piezas especiales es el diseño de problemas, con todo un subgénero de problemas dedicado a explorar sus posibilidades.

2.1.4 Organismos de regulación

Dado lo extendido que se encuentra el juego, ha sido necesario crear organismos que establezcan las reglas de juego a nivel mundial, pues de otra forma sería muy complicado organizar campeonatos entre naciones. Los dos principales organismos son la FIDE y la ICCF, aunque cada uno de ellos cuenta con entidades menores encargadas de controlar regiones o países de forma local. Tanto la FIDE como la ICCF organizan regularmente campeonatos, mantienen una clasificación de los jugadores afiliados a ellas, forman árbitros y establecen una serie de normas para la obtención de títulos, entre ellos el de campeón del mundo.

2.1.4.1 FIDE

La *Fédération Internationale des Échecs* fue fundada el 20 de Julio de 1924 en París y es hoy reconocida como la federación encargada de regular los campeonatos de ajedrez, así como las Olimpiadas de Ajedrez. Como máximo órgano regulador del deporte, nombra también al Campeón del Mundo, título que se conserva durante tres años, cuando debe defenderse frente al ganador del Torneo de Candidatos. En la actualidad el título lo tiene Viswanathan Anand, que tendrá que defenderlo en el 2012 frente a Boris Gelfand.

Además del título de campeón del mundo, la FIDE otorga los títulos de Maestro Internacional y

3 http://chesstempo.com/

4 http://www.chessgames.com/

Gran Maestro a aquéllos cuyo juego resulta destacable. Actualmente se han fijado una serie de normas que deben cumplirse para recibir cada uno de éstos títulos, siendo el título de Maestro Internacional requerido para alcanzar el de Gran Maestro.

También mantiene una clasificación de los jugadores federados, en función de su puntuación Elo (ver abajo), en categorías global, femenina, junior global y femenina junior. En Julio de 2011, los primeros clasificados en las categorías global y femenina eran Magnus Carlsen y Judit Polgar[5], respectivamente. Estas clasificaciones se actualizan mensualmente.

2.1.4.2 ICCF

La *International Correspondence Chess Federation* cumple el mismo papel que la FIDE, pero limitándose a los jugadores que practican el ajedrez por correspondencia o correo electrónico. Al igual que la FIDE, otorga los títulos de Maestro de Ajedrez por Correspondencia y Maestro Internacional en Ajedrez por Correspondencia, reconocidos por la FIDE, y mantiene una clasificación de los jugadores activos.

2.1.5 Clasificación de jugadores

Como en todo entorno competitivo, el mundo del ajedrez usó durante siglos clasificaciones para destacar a los mejores jugadores. Los métodos utilizados eran varios y no exentos de problemas, hasta que a mediados del siglo XX Arpad Elo creó el que se convertiría en el estándar oficial.

2.1.5.1 Elo

Creado por Arpad Elo, es probablemente el sistema más extendido de evaluación estadística en juegos de todo tipo, como Go o el sistema de juego en línea Xbox Live, a pesar de ser creado con el ajedrez en mente. Esto se debe a que es fácil de comprender y calcular, además de ser una buena base para la evaluación de jugadores individuales.

El valor Elo asociado a un participante se calcula en base al resultado de una partida entre dos jugadores, cuantificado en función de la diferencia de puntos entre ellos antes de empezar la partida. A mayor diferencia de puntos, mayor repercusión tendrá el resultado en las respectivas puntuaciones. La idea es que una derrota contra un jugador de mayor calidad no debería penalizar tanto como una derrota contra otro mucho peor, y a la inversa para las victorias y los empates.

5 http://ratings.fide.com/top.phtml?list=men, http://ratings.fide.com/top.phtml?list=women

Para calcular la modificación del Elo tras un encuentro, primero se determinan las probabilidades que tiene cada uno de los jugadores de ganar al rival. Siguiendo el estilo del ajedrez, un empate equivale a media victoria y media derrota para cada jugador.

Para dos jugadores A y B, con valores Elo R_A y R_B, la estimación de opciones de victoria para cada uno, E_A y E_B, se calcula con las fórmulas:

$$E_A = \frac{1}{1+10^{(R_B - R_A)/400}}$$

$$E_B = \frac{1}{1+10^{(R_A - R_B)/400}}$$

Desarrollando la fórmula se llega a que este sistema estima que la probabilidad de derrotar a un oponente aumenta 10 veces por cada 400 puntos de diferencia sobre él o ella.

Tras el encuentro, dado el porcentaje de victorias obtenido realmente, S_A y S_B, la nueva puntuación, R'_A y R'_B, se calcula con:

$$R'_A = R_A + K(S_A - E_A)$$

$$R'_B = R_B + K(S_B - E_B)$$

Donde K es el factor de ajuste lineal planteado por Elo, que le dio un valor de 10. Cuanto mayor sea K, más pronunciados serán los cambios tras resultados no esperados.

La FIDE aceptó el sistema de evaluación Elo en 1970, definiendo distintos tramos, según la experiencia de los jugadores, en un intento de asegurar que los buenos jugadores alcanzan posiciones altas rápido, pero se estabilizan posteriormente. Los jugadores de la FIDE se evalúan con:

- $K = 25$ para jugadores que hayan participado en menos de 30 eventos.
- $K = 15$ para jugadores que no han llegado a 2400 puntos, pero sí han jugado 30 eventos.
- $K = 10$ para jugadores con más de 30 eventos que han alcanzado alguna vez los 2400 puntos.

A pesar de su éxito, debido a lo que se consideran errores en las estimaciones de su creador, son pocas las organizaciones que lo usan en su forma original. Algunas de las modificaciones se limitan, como en el caso la FIDE, a cambiar los valores de K, mientras que otras utilizan distribuciones de probabilidad distintas a la de Elo, alterando el cálculo de la probabilidad de victoria.

2.1.5.2 Glicko

Glicko [GLIC 4] es una actualización de Elo que introduce un factor de fiabilidad en forma de desviación de la clasificación *RD*, de *Ratings Deviation*. Para un jugador, este valor disminuye con cada partida jugada, pero aumenta conforme pasa el tiempo sin registrar nuevas entradas.

Usando *RD*, se dice que la fuerza *F* de un jugador con una puntuación Elo *E* es:

$F = E \pm (2 \times RD)$ con confianza del 95%.

Para jugadores sin clasificación anterior o que no han jugado en mucho tiempo, se establece un valor de *RD* de 350, que sólo puede disminuir o mantenerse.

Existe una segunda versión de Glicko, Glicko-2 [GLI2 5], que añade un factor de volatilidad a la estimación, teniendo en cuenta la consistencia en los resultados de un jugador.

2.1.6 Notación de partidas

Con el objetivo de poder almacenar, compartir y analizar partidas se crearon una serie de formatos que permitían guardar la información relevante. En las últimas décadas, este tipo de formatos se han pensado principalmente con el objeto de ser usadas por sistemas informáticos.

2.1.6.1 Forsyth-Edwards Notation

Más conocido como FEN, este formato, en vez de guardar los movimientos de una partida, se centra en almacenar una posición del tablero en particular. Aunque almacena toda la información relevante, el hecho de que se limite a una única posición hace imposible determinar si es posible declarar tablas por repetición triple. Para ello sería necesario contar con todas las posiciones de la partida, momento en el cual es preferible recurrir a cualquier formato que almacene movimientos. Sí es posible, por otra parte, determinar la posibilidad de empate por la regla de los 50 movimientos.

Su objetivo es poder continuar una partida desde un punto en particular, sin importar lo que haya ocurrido antes. Se puede usar para problemas, entrenamiento o estudio.

En FEN toda la información se guarda en una línea, con cada campo separado del siguiente por un espacio. Las piezas son identificadas por la inicial en inglés, con *N* haciendo referencia al caballo. Se usan minúsculas para las piezas negras y mayúsculas para las blancas.

En orden, los campos son los siguientes:

- Posiciones en el tablero: Empezando en *h1*, por líneas, hasta *a8*. Si hay una pieza en la casilla correspondiente, se indica. Si no, se indica el número de casillas desocupadas consecutivas en la fila. Las filas se separan con una barra (/).
- Turno: una *w* o *b*, según si el turno es de blancas o negras, respectivamente.
- Enroque: posiciones con las que es posible enrocar por parte de cada rey. Primero se indican las opciones para blancas y luego para negras. Se usa el identificador de rey (K) o reina (Q), en mayúsculas o minúsculas, si es posible enrocar del lado de rey o reina, respectivamente. Si ningún rey puede enrocar, se escribe un guion (-).
- Captura al paso: si en el turno anterior un peón avanzó dos casillas, se almacena la posición en que se le podría capturar. No importa que no haya un peón en la posición necesaria para realizar la captura. Si no hubo tal avance, se escribe un guion (-).
- Contador de jugadas: número de jugadas (medio movimiento) que se han ejecutado desde la última captura o movimiento de peón. Necesario para declarar tablas con la regla de los 50 movimientos.
- Contador de movimiento: número de movimientos (dos jugadas) transcurridos desde el principio de la partida.

Para la partida **1.** e4 c5 **2.** Nf3 Nf6 **3.** Rg1, por ejemplo, la primera y última entradas serían:

Figura 1: 1. e4

```
rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR b KQkq e3 0 1
```

Figura 2: 1. e4 c5 2. Nf3 Nf6 3. Rg1

```
rnbqkb1r/pp1ppppp/5n2/2p5/4P3/5N2/PPPP1PPP/RNBQKBR1 b Qkq - 3 3
```

2.1.6.2 Portable Game Notation

El formato más popular en la actualidad es el PGN, creado en los primeros tiempos de Internet con el objetivo de facilitar el intercambio de partidas. PGN incluye dos formatos similares, preparados para, específicamente, importar y exportar partidas. El primero es algo menos estricto, para soportar partidas escritas a mano, mientras que el segundo detalla exactamente cada aspecto a tener en cuenta. Dos programas que usen el formato de exportación para la misma partida en la misma máquina están obligados a generar dos archivos binarios idénticos.

Una partida en PGN empieza describiendo propiedades de la misma. Como mínimo debe incluir: evento, lugar, fecha, ronda, quién juega con blancas y negras y el resultado de la partida. Existen otras propiedades de las que se puede prescindir.

Tras los pares de datos se añade el conjunto de movimientos, en notación algebraica (ver [FIDE 6]), seguida del resultado de la partida, una vez más, pero sólo si la partida ha terminado.

2.2 Ajedrez e informática

Si bien es cierto que la popularización de la informática, Internet y otras tecnologías modernas han cambiado el mundo en que vivimos, no es lo mismo con la inmensa mayoría de los deportes. Salvo mejoras en los estudios de materiales, simulaciones, instrumentación y, por supuesto, una revolución en la cobertura informativa, los deportes físicos, de motor y demás siguen en gran medida disputándose de igual forma que hace 50 años.

No es el caso del ajedrez. Aunque las reglas han permanecido estables, las estrategias consideradas óptimas, así como el conocimiento del juego en sí, han variado tanto que cada década puede considerarse una nueva escuela de pensamiento. Jugadas que hace 20 años provocaban aplausos son ahora analizadas con desdén, o posiciones poco favorables según maestros de los 70 son ahora explotadas de formas impensables para ellos [BRIT 1].

Las principales causas de esta revolución son tres. En primer lugar, el uso de software informático para estudiar partidas, variantes y analizar gran cantidad de posibilidades en poco tiempo. La percepción del valor de ciertas posiciones, piezas o estrategias es ahora mucho más precisa gracias a esto. Segundo, la posibilidad de jugar cientos de partidas al día, sin riesgo ni excesivas penalizaciones, a través de internet. Esto permite desarrollar un juego mucho más instintivo, basado en experiencias reales de juego, en lugar de simulaciones y estudio, lo que según algunos jugadores les ayuda a evitar decisiones pobres. Y, tercero, la existencia de grandes bases de datos de partidas completas o parciales, con fácil acceso y ayudas para el filtrado. Estudiar finales de juego es ahora una tarea mucho menos ardua que hace sólo 15 años, circunstancia de la que todo jugador profesional se aprovecha.

Pasaremos ahora a estudiar en detalle algunos de éstos puntos, dada la íntima relación que guardan con el proyecto de ChessAsin.

2.2.1 Inteligencia artificial

Desde mediados de los 60, con el despegue de las tecnologías informáticas, empezaron a llevarse a cabo intentos de crear programas capaces de jugar al ajedrez, entre otros muchos juegos. Aunque los primeros tiempos no fueron nada prometedores, dieron lugar a la investigación de técnicas que, tras ser refinadas, permitirían a equipos posteriores aprovechar mejor sus recursos.

Tras décadas de desarrollo, la primera máquina en conseguir derrotar a un campeón del mundo fue el famoso Deep Blue, desarrollado por IBM, que contaba con hardware especializado para evaluar posiciones de ajedrez. Con el aumento de la capacidad de cálculo y el perfeccionamiento de las técnicas implementadas, hoy día existen varios programas, incluso de código libre, capaces de derrotar a grandes maestros cuando se ejecutan en equipos domésticos.

En la actualidad, los programas de ajedrez más potentes son Houdini, Rybka, Critter y Stockfish[6], algunos de ellos con versiones para dispositivos móviles.

6 Consultado en http://www.computerchess.org.uk/ccrl/4040/, a 2 de Noviembre de 2011.

2.2.2 Bases de datos

Antiguamente las partidas o jugadas interesantes se encontraban clasificadas en libros especializados, normalmente de gran tamaño, que sólo jugadores de gran solvencia tenían. Estos libros eran referencia obligada para cualquiera que deseara prosperar como jugador, ya fuera como aficionado o profesional, pero suponían un problema de transporte y consulta.

En la actualidad existen grandes bases de datos especializadas en juegos, aperturas o finales de partida, que permiten la consulta con interfaces varias. Además, estas bases de datos no tienen por qué estar en posesión del jugador, sino que pueden ser almacenadas en servidores para consultas remotas. Debido al gran tamaño de algunas de ellas, que pueden llegar a varios terabytes, ésta es una gran ventaja.

2.2.2.1 Bases de datos de juegos

Éstas son bases de datos que contienen juegos completos, ya sean clásicos, de campeonatos, jugados en un servidor en particular, etc. Las más solicitadas son aquéllas con partidas de grandes maestros o de juegos estratégicamente interesantes.

2.2.2.2 Bases de datos de aperturas

Con la importancia que las aperturas y defensas tienen en el ajedrez, es fundamental conocerlas bien, estudiándolas a fondo. Bases de datos comentadas existen para prácticamente todas las variantes, dentro de determinados parámetros.

2.2.2.3 Bases de datos de finales de partida

Cuando al final de una partida quedan pocas piezas en juego, puede ser determinante encontrar en poco tiempo la combinación de movimientos que pueda provocar un mate o forzar un empate. Según las piezas que haya en juego las opciones cambian por completo, por lo que es imposible para un ser humano recordar todas las posibilidades, más allá de los finales más comunes.

Existen bases de datos de finales de 4 piezas o 5, o combinaciones específicas de más piezas, pero. debido a la complejidad de analizar y almacenar todas las variantes, no es habitual encontrarlas de más. Es importante tener en cuenta que el número y posición de los peones en juego puede cambiar completamente finales de partida con piezas semejantes.

2.2.3 Aplicaciones de juego populares

Dada la popularidad del ajedrez, junto con su relativa simplicidad, no es raro descubrir que desde el principio de la informática han existido programas dedicados a jugarlo o analizarlo. La evolución de los mismos ha ido unida a las capacidades de los terminales, principalmente relacionados con mejoras en las interfaces o la potencia de cálculo, hasta la aparición de las conexiones remotas, que abrió todo un nuevo campo que explorar.

2.2.3.1 Clientes de ordenador

Históricamente, éstos han sido programas de juego a dobles o contra una mediocre inteligencia artificial en local. La mayor innovación entre versiones de una misma franquicia solía encontrarse en nuevos gráficos y mejor inteligencia artificial.

Con el paso del tiempo empezaron a surgir algunas versiones que permitían jugar a través de redes o de correo. También se han terminado popularizando los programas que hacen de interfaz para aplicaciones de consola o puramente lógicas, de forma que cada usuario pudiera escoger la presentación que prefiriese, manteniendo la posibilidad de jugar contra rivales con diferentes preferencias. Uno de los sistemas más conocidos es el de *XBoard*, que permite jugar por correo electrónico, o usando uno de varios motores de juego o servidores de ajedrez.

Los clientes más famosos en la actualidad son *Fritz*/*Deep Fritz*, *ChessTiger* y el ya mencionado *XBoard*. El punto fuerte de los dos primeros se encuentra en su elaborada inteligencia artificial y en las ayudas que ofrecen para el aprendizaje. El tercero, por otra parte, basa su popularidad en la gran cantidad de modos de juego que soporta, incluyendo variantes del ajedrez o las versiones india, china y japonesa, así como múltiples módulos para jugar en línea o con diversas interfaces gráficas.

2.2.3.2 Clientes en páginas web

Una de las modalidades más populares de juego son las interfaces web. Existen miles de páginas, cada una con distintos servicios, como juego en tiempo real, por correspondencia, problemas o características propias de redes sociales. Normalmente se juega a través de una interfaz programada en JavaScript, Flash u otra alternativa, aunque algunos cuentan con clientes externos que hacen de intermediarios entre el usuario y la web.

Normalmente requieren una cuenta, para poder seguir el progreso de cada jugador, y permiten observar, analizar y comentar partidas de otros jugadores. Algunas páginas incluyen también

servicios de entrenamiento, problemas o consulta.

Las más populares son FICS (*Free Internet Chess Server)*[7] y ChessTempo[8], aunque también existen comunidades menores de pago, pero muy activas, como Red Hot Chess[9], con organizaciones de campeonatos y equipos que participan en torneos oficiales.

2.3 Plataformas móviles

En la última década la revolución informática ha dado el salto a los dispositivos móviles, como respuesta al progresivo estancamiento de los ordenadores de sobremesa. La evolución más notable ha afectado a los teléfonos móviles, que hoy día rivalizan en potencia, prestaciones y mercado con equipos domésticos o videoconsolas, encontrándose entre los artículos tecnológicos más vendidos y con expectativas de continuado crecimiento.

Esta revolución ha creado también un nuevo mercado para los desarrolladores de software, puesto que los dispositivos móviles han venido acompañados por plataformas de distribución digitales y globales, con reducidos costes de publicación y fácil acceso para los usuarios finales. Esto permite que pequeños equipos creen y comercialicen aplicaciones con tiempos y costes de desarrollo mucho menores de lo habitual, a la vez que con más facilidades para cubrir gastos y generar beneficios.

En los siguientes apartados analizaremos las plataformas de mayor éxito entre usuarios no profesionales, prestando especial atención a las facilidades que ofrecen a los creadores de contenidos.

2.3.1 Sistemas operativos

En la actualidad las principales plataformas a nivel de usuario son las siguientes.

7 http://www.freechess.org

8 http://chesstempo.com

9 http://www.redhotchess.com

2.3.1.1 Symbian

La primera compañía en popularizar los teléfonos potentes y con aplicaciones desarrolladas por usuarios fue Symbian, apoyado por Nokia, quien más tarde pasaría a ser el propietario del sistema operativo. A pesar de su control del mercado, las dificultades que presentaba para el desarrollo o uso, así como el estancamiento de su funcionalidad y problemas de retrocompatibilidad, hizo que perdiera terreno rápidamente frente a alternativas como BlackBerry o iPhone [ORLO 7].

Debido a la situación en que se encuentra Nokia y su reciente dedicación exclusiva a Windows Phone 7, Symbian se halla en una situación comprometida. En breve pasará a ser desarrollado por Accenture[10], bajo un acuerdo con Nokia, pero se desconocen los planes que ésta pueda tener para la plataforma a largo plazo. Debido a esto, no analizaremos Symbian en mayor detalle.

2.3.1.2 BlackBerry

Creado, distribuido y controlado por RIM, BlackBerry es un sistema operativo enfocado a un mercado de profesionales con demandas muy específicas. Por ello, RIM ofrece servicios especiales de gestión de correos o contenidos, adaptados a las necesidades de empresas. A consecuencia de esto, BlackBerry, a pesar de su éxito en entornos empresariales, no ha conseguido introducirse en el mercado de usuarios domésticos.

En los últimos años BlackBerry está además perdiendo cuota de mercado en su propio campo respecto a iOS o Android, a medida que estas plataformas maduran y ofrecen servicios alternativos a los de RIM. Dado su limitado impacto en aplicaciones de usuario, tampoco tendremos en cuenta BlackBerry en los siguientes análisis.

2.3.1.3 iOS

Introducida por Apple para sus teléfonos móviles iPhone y reproductores iPod, supuso el definitivo impulso de las plataformas móviles inteligentes. Esta plataforma está completamente controlada por Apple, siendo ellos los únicos que desarrollan y venden sus productos. Aunque esto facilita considerablemente el desarrollo de aplicaciones, al limitar el número de configuraciones *hardware* posibles a un conjunto conocido, otorga un control a Apple sobre qué se publica en su plataforma que preocupa a muchos.

Existen principalmente tres dispositivos equipados con el sistema operativo iOS: iPod Touch,

10 http://www.engadget.com/2011/06/22/nokia-and-accenture-seal-symbian-handover-deal/

iPhone y iPad, habiendo recibido cada uno de ellos al menos una actualización de hardware. En total, el número de dispositivos existentes no pasa de la docena, y el sistema operativo ofrece mecanismos para detectar las características del aparato. Esto permite adaptar las aplicaciones a las versiones menos potentes, o simplemente declararlas no compatibles, sin confundir a los usuarios.

2.3.1.4 Android

Controlado por la Open Handset Alliance, encabezada por Google, es desde su aparición el principal contendiente de iOS. Está basado en una implementación de código abierto de Java montada sobre Linux. Su gran popularidad entre los fabricantes de móviles se debe a su distribución gratuita, lo que abarata costes. Por otra parte, su gran debilidad es la actual fragmentación del ecosistema, con dispositivos relativamente nuevos relegados a un segundo plano al no ser actualizados a versiones más recientes del sistema operativo. El problema llegó a ser de tales dimensiones que Google ha obligado a los fabricantes que usen Android a comprometerse a actualizar sus móviles durante, al menos, su primer año de vida.

Desde el punto de vista del hardware, Android no especifica ningún requisito particular, por lo que las características de los dispositivos varían ampliamente. Esto complica el desarrollo de aplicaciones, pues es necesario comprobar una variedad de configuraciones considerable.

2.3.1.5 Windows Phone 7

El último sistema operativo para móviles en entrar en la competición, Windows Phone 7, supuso una importante renovación de la anterior oferta de Microsoft, Windows Mobile 6.5. Es en la actualidad la plataforma móvil con mayor índice de crecimiento, lo que se espera que se mantenga, pues progresivamente aumenta el número de dispositivos disponibles. A pesar de estar controlada por Microsoft, este sistema operativo puede ser instalado por todos los fabricantes de móviles que deseen utilizarlo. Esto lo sitúa a caballo entre la situación de iOS y Android. Para evitar la coexistencia de versiones anticuadas, Microsoft obliga a los fabricantes a actualizar con cierta regularidad sus dispositivos. Aunque la implementación actual del sistema no acaba del todo con la fragmentación, sí ha conseguido evitar una situación como la que afectaba a Android en sus primeros años.

En lo referido al *hardware*, Microsoft especifica claramente los requisitos que un móvil debe cumplir para poder usar WP7. Así pues, a pesar de la gran cantidad de móviles disponibles, todos

tienen las mismas características, facilitando el desarrollo de aplicaciones compatibles con todos ellos. Versiones futuras de Windows Phone incluirán una nueva lista de requisitos, para que las aplicaciones puedan adaptarse a cada revisión sin problemas.

2.3.2 Herramientas de desarrollo

A la hora de desarrollar aplicaciones para móviles, una de las primeras cosas a tener en cuenta son las herramientas y requisitos asociados a cada plataforma. En este apartado analizaremos estos factores para iOS, Android y WP7.

2.3.2.1 iOS

Aunque se permite usar lenguajes de scripting en situaciones específicas, los lenguajes base permitidos por Apple son C y Objective-C, una ampliación de C que añade, principalmente, soporte para objetos y un sistema de paso de mensajes por cadenas. Debido a su popularidad, existen gran cantidad de bibliotecas que añaden funcionalidad a estos lenguajes, o perfeccionan la ya existente, así como multitud de entornos integrados de desarrollo. La principal de éstas ampliaciones es COCOA, que se especializa en la creación de aplicaciones gráficas sobre Objective-C.

Las APIs ofrecidas por Apple para sus plataformas son considerablemente completas y son ampliamente conocidas, lo que facilita la resolución de problemas. Aún así, la dependencia en lenguajes de desarrollo complejos y anticuados dificulta considerablemente el desarrollo.

Otra limitación a tener en cuenta es la obligación de compilar y probar las aplicaciones de iOS en sistemas operativos MacOS, con nuevas versiones del sistema operativo móvil eliminando soporte de versiones antiguas de MacOS. Para desarrolladores sin un equipo de Apple esto puede suponer un desembolso considerable, teniendo en cuenta los presupuestos en que se desenvuelven los programadores independientes.

2.3.2.2 Android

Montada sobre la plataforma Java, es éste el lenguaje de programación a utilizar a la hora de desarrollar aplicaciones. Existen además intérpretes de lenguajes de script para múltiples lenguajes, como Python o Perl, aparte de proyectos más ambiciosos, como MonoDroid, una adaptación de Mono, la versión libre de .NET, a Android. Para secciones en las que la velocidad de ejecución es crírica se ha creado el Android Native Development Kit[11] (NDK), que permite integrar secciones de código C/C++ en las aplicaciones Java.

La madurez de Java, así como la gran cantidad de bibliotecas externas existentes para el lenguaje, hacen que Android sea una alternativa interesante. Una de las principales mejoras que Google ha introducido en su versión de Java es el soporte para la definición de interfaces gráficas independientes por completo del código, usando archivos de configuración XML parecidos en objetivos y mecanismos a las *CSS* de HTML.

Android permite además desarrollar aplicaciones desde cualquier entorno capaz de ejecutar Java, incluyendo MacOS, Windows y Linux, con Eclipse como el entorno de desarrollo recomendado.

2.3.2.3 Windows Phone 7

Microsoft ha montado WP7 sobre la plataforma .NET, ya establecida como la preferida para el desarrollo de aplicaciones en Windows. Debido a las limitadas prestaciones del móvil no está disponible la plataforma .NET completa, sino un subconjunto de ésta, conocido como *.NET Compact Framework*, que se usa en dispositivos incrustados y la consola de videojuegos Xbox 360. De todos los lenguajes de .NET, WP7 soporta principalmente C#, aunque también es posible utilizar Visual Basic.Net y F#.

De las facilidades que .NET ofrece a los desarrolladores de WP7 destacaremos tres:

- Silverlight: a su vez un subconjunto de Windows Presentation Foundation, Silverlight ofrece funcionalidad para la creación y manipulación de interfaces gráficas 2D, en un estilo similar al de Adobe Flash.
- XNA: conjunto de bibliotecas de nivel medio-alto orientadas al desarrollo de videojuegos, montado sobre una arquitectura jerárquica. Para Xbox 360 y WP7 sólo soporta aplicaciones escritas en C#.

11 http://developer.android.com/sdk/ndk/index.html

- Windows Communication Foundation (WCF): es un framework de comunicación de aplicaciones por red, con especial énfasis en servicios web y estructuras cliente-servidor. Para WP7 toda comunicación a través de WCF debe ser obligatoriamente asíncrona, y cuenta además, por depender del *Compact Framwork*, con limitaciones adicionales en los mecanismos de comunicación disponibles.

Existen múltiples bibliotecas externas a Microsoft para añadir funcionalidad a .NET y WP7, y Microsoft mantiene CodePlex como repositorio de código público para sus plataformas, la mayoría publicado bajo la licencia de código libre Microsoft Public License (MS-PL)[12].

El desarrollo de aplicaciones para WP7 puede realizarse desde el entorno de desarrollo Visual Studio 2010, en cualquiera de sus versiones, incluyendo Express, que es gratuita. Está limitado a sistemas Windows, con soporte parcial para Windows XP.

2.3.3 Plataformas de distribución

Desde la aparición de Symbian, una de las principales quejas de los usuarios era la dificultad de encontrar aplicaciones útiles y fiables. En general, el usuario debía ser capaz de encontrarlas e instalarlas en su móvil, tarea impensable para la mayoría de clientes de la plataforma.

Para solventar este problema, Nokia creó servicios como *Widsets* o *Download!*, que más tarde sustituiría con el más completo *Ovi Store*[13], exclusivo para dispositivos de la compañía.

Para los siguientes apartados usaremos como referencia las publicaciones de Distimo [DIST 8 April] y [DIST 9 May], con datos de Abril y Mayo de 2011, respectivamente.

2.3.3.1 Apple App Store

A pesar del éxito de Nokia con sus plataformas de distribución, fue Apple la compañía que popularizó definitivamente estos sistemas de venta para dispositivos móviles. El número de aplicaciones disponibles en Abril en esta plataforma se encontraba en 333.214, con un crecimiento desde el mes anterior del 6%.

Para publicar en App Store se debe ser un desarrollador registrado, pagando una cuota anual y con los beneficios de ventas divididos al 30% para Apple y 70% para el desarrollador. Apple también cobra cuotas por suscripciones hechas desde su plataforma, así como por otros servicios.

12 http://www.opensource.org/licenses/MS-PL

13 http://mobile.engadget.com/2009/04/24/nokia-rolling-widsets-into-ovi-store

Toda aplicación subida al App Store debe pasar un proceso de aprobación por parte de Apple, y de aquí surge la principal queja de los desarrolladores: las normas que deben cumplir sus programas no se encuentran especificadas claramente. Solamente se publican guías generales, por lo que es imposible saber si una aplicación será rechazada o no. Además Apple ha cambiado las normas sin previo aviso varias veces, normalmente para evitar la entrada de aplicaciones que compitiesen con las propias de la compañía, o plataformas de desarrollo como Adobe Air. Estos factores de incertidumbre han hecho que muchos desarrolladores se planteen migrar a otros servidores de aplicaciones.

2.3.3.2 Android Market

Controlado por Google, el Android Market cumple la misma función que el App Store para iOS. Contaba en Abril con 206.143 aplicaciones, con un crecimiento del 16%, aunque más del 65% de ellas son gratuitas, frente al 30% aproximado de Apple. Esto se puede deber a que, según estudios de Mayo de 2011, sólo el 20% de las aplicaciones de pago disponibles en el Android Market han sido compradas más de 100 veces. Muchas de las aplicaciones gratuitas, por tanto, generan beneficios por medio de publicidad, usando las APIs que Google ofrece para ello u otras similares. Es también normal que una aplicación tenga una versión gratuita y otra de pago, sin publicidad o con funcionalidad añadida.

Google también revisa las aplicaciones, pero siempre tras su publicación, lo que ha hecho que se encuentren varias aplicaciones malware[14]. Esto preocupa a los desarrolladores, pues los usuarios son menos dados a descargar aplicaciones o pagar por ellas, por temor a comprometer sus dispositivos.

2.3.3.3 Amazon AppStore

Es una alternativa que Amazon ofrece para los usuarios de Android. El proceso de envío de aplicaciones es distinto y sus medidas de seguridad mejores que las de Android Market, pues las aplicaciones son analizadas antes de ponerse a la venta.

Las ventajas que ofrece este sistema frente al de Google son, aparte de la mejora en la seguridad, herramientas de búsqueda más adecuadas para descubrir buenas aplicaciones. Además incluye secciones especiales para nuevas aplicaciones, dándoles mejor visibilidad de la que tendrían en el Android Market.

14 http://edition.cnn.com/2011/TECH/mobile/03/02/google.malware.andriod/

Pero también existen problemas que han causado la retirada de aplicaciones del AppStore[15] y enérgicas protestas por parte de asociaciones de desarrolladores[16]. En primer lugar, Amazon se reserva el derecho a cambiar los precios de aplicaciones en periodos de oferta sin consultar a los desarrolladores, reduciendo sus beneficios. Por otra parte, Amazon no ofrece información específica que permita saber al comprador si una aplicación funcionará o no en su dispositivo, lo que puede generar confusiones y peores puntuaciones para aplicaciones especialmente exigentes.

Por supuesto, el mayor obstáculo para la distribución que presenta Amazon AppStore es su escasa visibilidad frente a la opción de Google, que está disponible de base en los móviles Android.

2.3.3.4 Windows MarketPlace para móviles

Como era de esperar, Microsoft presentó un mecanismo de distribución de aplicaciones cuando anunció WP7. Dada su corta existencia, en Abril era la plataforma con menor número de aplicaciones, con 11.731, pero con mayor crecimiento, 38%, lo que podría situarla rápidamente entre las más populares. Un detalle a destacar es que Microsoft ha publicado en los últimos meses bibliotecas que facilitan la conversión de aplicaciones de iOS o Android a Windows Phone 7, por medio de APIs que simulan las de dichas plataformas y guías para desarrolladores acostumbrados a ellas. Debido a esto se espera que el crecimiento futuro de aplicaciones en el MarketPlace esté por encima de ese 38%.

Para evitar los problemas que Apple y Google han encontrado en sus procesos de aprobación, Microsoft publicó los requisitos exactos que una aplicación debía cumplir para entrar en su MarketPlace varios meses antes de que la propia plataforma existiese. Además, cuando éstos son actualizados se avisa a los desarrolladores con al menos un mes de antelación, para que puedan adaptar las aplicaciones casi finalizadas antes de intentar publicarlas.

Pero MarketPlace no está libre de problemas, como es habitual: aunque Microsoft está dando pasos para resolverlo, es innegable que las aplicaciones de WP7 son fácilmente pirateables. Otra barrera que afecta a esta plataforma es que los parches de las aplicaciones pasan el mismo proceso de verificación que las aplicaciones, lo que puede retrasar la instalación de parches de seguridad en las aplicaciones de los usuarios hasta un mes. Por supuesto, esto sirve para impedir la introducción de

15 http://arstechnica.com/gadgets/news/2011/07/amazon-appstore-game-developer-pulls-app-highlights-problems.ars

16 http://igdaboard.wordpress.com/2011/04/14/important-advisory-about-amazon%e2%80%99s-appstore-distribution-terms-2/

malware por medio de actualizaciones.

2.3.4 Aplicaciones de ajedrez para móviles

Dada la popularidad del ajedrez, no es de extrañar que cada plataforma móvil cuente con gran cantidad de aplicaciones para jugar. A continuación daremos un repaso a las alternativas existentes en cada plataforma.

2.3.4.1 iOS

Siendo la primera plataforma en atraer grandes cantidades de desarrolladores, iPhone es sin duda el móvil con mayor cantidad de aplicaciones para ajedrecistas. Dadas las posibilidades que ofrece Apple para el desarrollo de aplicaciones, algunas de ellas, principalmente las más antiguas, están preparadas para jugarse a través de Safari, el navegador web del móvil, en páginas preparadas específicamente para la pequeña pantalla. Con el paso del tiempo empezaron a surgir aplicaciones específicas para iPhone, hasta el centenar que se pueden encontrar en el App Store en la actualidad.

La mayoría de las aplicaciones están limitadas a enfrentamientos contra la máquina, con inteligencias artificiales de dificultad adaptable. En esta modalidad destacan Shredder Chess, Fritz Chess y tChess Pro, con versiones para móviles de sus potentes motores de juego. Todas, en general, permiten partidas jugador contra jugador, en el mismo móvil o, en algunos casos, a través de BlueTooth. Son pocas las que permiten jugar al ajedrez en línea y muchas menos las que permiten hacerlo por correspondencia o con algún sistema parecido. Las pocas que dan esta opción suelen utilizar FICS o ICC como servidores de juego.

Otro de los modos de juego poco explotados son los problemas de ajedrez, con menos de una decena de aplicaciones con esta opción. Entre ellas destaca Caissa Chess, que combina una inteligencia artificial decente para el juego normal y un buen sistema de puzzles.

En general, todas las aplicaciones suelen mostrar la notación algebraica de la partida en curso, así como ver la evolución de la partida avanzando y retrocediendo por los movimientos ejecutados. Sin embargo, pocas permiten estudiar partidas famosas o de otros jugadores del sistema. Otra facilidad relativamente común es la posibilidad de usar una guía de juego, con bibliotecas de salidas, recomendaciones de movimientos o avisos contra malas jugadas.

2.3.4.2 Android

Basta dedicar un poco de tiempo para encontrar una decena de juegos de ajedrez disponibles para Android. Al igual que con iOS, los hay de todos tipos, principalmente debido a que muchas de las aplicaciones son versiones adaptadas del iPhone. Muchas de ellas, como es habitual en Android, tienen una versión gratuita y otra de pago, aunque abundan las de precio reducido. Entre las propias de Android destaca Chess Buddies, con una comunidad muy activa.

En general, las características que se dan en iPhone se encuentra repetidas en Android, aunque con la falta de algunas de las aplicaciones mejor consideradas del móvil de Apple.

2.3.4.3 Windows Phone 7

A finales de Julio del 2011, en el Windows MarketPlace se podían encontrar apenas 5 aplicaciones para jugar al ajedrez, estando cuatro de ellas limitadas a jugar en el propio móvil contra la máquina o un rival, o por BlueTooth. Sólo PocketGrandMaster permite jugar en línea, usando los servicios de FICS o ICC. El apartado de problemas de ajedrez serios está, por otra parte, prácticamente desierto entre las ofertas de Microsoft.

Con tan poca competencia, parece claro que pronto empezarán a aparecer aplicaciones que llenen este hueco.

2.4 Conclusiones

Tras evaluar la popularidad del ajedrez hoy día, así como las diferentes plataformas móviles y sus canales de distribución, podemos llegar a las siguientes conclusiones:

1. El ajedrez sigue siendo uno de los deportes más populares del mundo, con millones de jugadores distribuidos por todo el mundo.
2. El juego a través de Internet desde plataformas digitales supera en usuarios y regularidad a las partidas sobre un tablero.
3. Las plataformas móviles están eclipsando a los ordenadores tradicionales como fuente de ingresos para pequeños desarrolladores, dada su ubicuidad y simples canales de distribución.
4. Existe un nicho aún poco explotado de aplicaciones de ajedrez para móviles que permitan jugar entre sistemas operativos distintos, y por ahora casi todas las aplicaciones usan servidores construidos con tecnologías antiguas, difíciles de mejorar.

5. La plataforma móvil más prometedora para nuestros intereses es Windows Phone 7, por ser la que mayor crecimiento está experimentando en la actualidad y en la que menos explotadas están las aplicaciones para ajedrecistas.

Una vez analizadas estas conclusiones, hemos decidido desarrollar un servidor de juego en línea de ajedrez, capaz de soportar múltiples mecanismos de comunicación y plataformas clientes. Lo desarrollaremos en algún lenguaje moderno, con el objetivo de facilitar la introducción de mejoras en el futuro, y con Windows Phone 7 como principal plataforma objetivo para el cliente.

Capítulo 3: Objetivos

El objetivo de este proyecto es implementar un sistema cliente-servidor que permita jugar al ajedrez a usuarios humanos, mediante las reglas de juego por correspondencia. El desarrollo estará dirigido a la plataforma móvil Windows Phone 7. A continuación pasaremos a describir los objetivos concretos para cada uno de los componentes del sistema.

3.1 Motivaciones del proyecto

Desde que empezaron a venderse ordenadores personales, toda nueva plataforma ha contado rápidamente con, al menos, una aplicación de ajedrez, que casi siempre demuestran ser un éxito. Tanto jugadores aficionados como expertos procuran tener una siempre a mano, en un formato más potente que lo que ofrece un simple juego con su tablero y fichas, por lo que no es extraño que dispositivos móviles, como teléfonos o PDAs, hayan tenido aún más éxito.

En el momento en que se planteó la posibilidad de iniciar el desarrollo, a finales de 2010, las dos principales plataformas para teléfonos móviles, iOS de Apple y Android de la Open Handset Alliance, tenían ya a disposición de sus usuarios varias aplicaciones para jugar al ajedrez. La mejor opción para introducirse en un mercado sin excesiva competencia sería la nueva plataforma de Microsoft, que saldría al mercado poco después.

Otra gran ventaja que ofrece Windows Phone 7 sobre la competencia son las facilidades que presenta a los desarrolladores: la plataforma .NET, junto con C# y un potente conjunto de bibliotecas.

Además, a título personal, dada mi inquietud por el desarrollo de juegos y los sistemas distribuidos, crear una arquitectura cliente-servidor que permitiera explorar las capacidades de la plataforma .NET resultaba muy interesante.

3.2 Objetivos

Con ChessAsin se pretende crear una arquitectura cliente-servidor con los siguientes componentes:

- Una base de datos MySQL capaz de almacenar y gestionar la información de usuarios y juegos.
- Biblioteca de acceso a la base de datos, programada en C#, con sus correspondientes mecanismos de seguridad y pruebas unitarias.

- Una implementación del ajedrez que permita desarrollar partidas enteras siguiendo las reglas del ajedrez por correspondencia.
- Un servidor para jugar al ajedrez asíncrono a través de Internet, montado sobre la arquitectura WCF de .NET y haciendo uso de la base de datos. Nos centraremos en la modalidad de juego por correspondencia, en lugar de juego en tiempo real, para permitir el uso de ChessAsin en dispositivos móviles, cuya conexión a Internet no es fiable al 100%.
- Un cliente de prueba desarrollado en XNA que permita identificar usuarios, crear y jugar partidas. Por limitaciones técnicas de WCF en la plataforma Windows Phone 7, cuya resolución caería fuera del alcance de este proyecto, el cliente se ha desarrollado para Windows, pero siguiendo una estructura que más tarde permita una rápida conversión a la plataforma móvil.

Capítulo 4: Desarrollo

En esta sección describiremos el proceso llevado a cabo para completar los principales subsistemas de ChessAsin.

En primer lugar expondremos los casos de uso y requisitos que definen el comportamiento de la aplicación, para más tarde analizar su efecto en el diseño de la base de datos y las librerías necesarias. Por último, repasaremos el proceso de implementación.

4.1 Análisis

La parte principal del sistema se encuentra en la interacción entre los clientes y el servidor. Cada uno de los clientes realizará peticiones a las que el servidor, tras evaluar su validez, responderá actualizando la base de datos, obteniendo datos de ella y enviándolos de vuelta al cliente. Aunque todas las peticiones se encuentran en el mismo servidor, hemos dividido los casos de uso según su funcionalidad.

4.1.1 Gestión de cuentas

En primer lugar, la funcionalidad que permite gestionar y usar cuentas. Toda interacción con el servidor debe iniciarse con una operación de inicio de sesión o creación de cuenta, pues cualquier petición desde un cliente no identificado es rechazada.

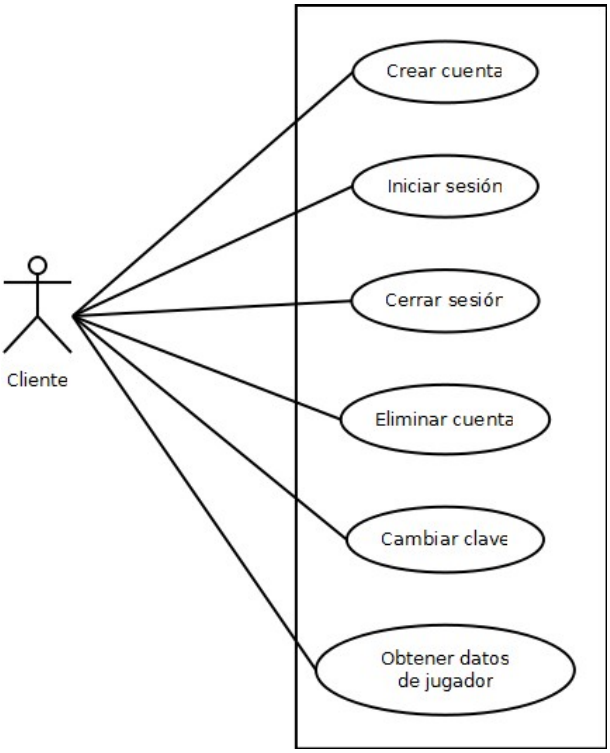

Figura 3: Casos de uso para cuentas.

4.1.2 Invitaciones y peticiones de rival

Para iniciar una partida debe encontrarse un rival, que puede ser conocido, en cuyo caso la solicitud se hace por nombre. Es posible consultar las invitaciones recibidas o creadas. El creador podrá además cancelar una invitación que ya no le interese mantener, mientras que el receptor podrá aceptarla o rechazarla. En caso de aceptar una invitación, una nueva partida es iniciada inmediatamente.

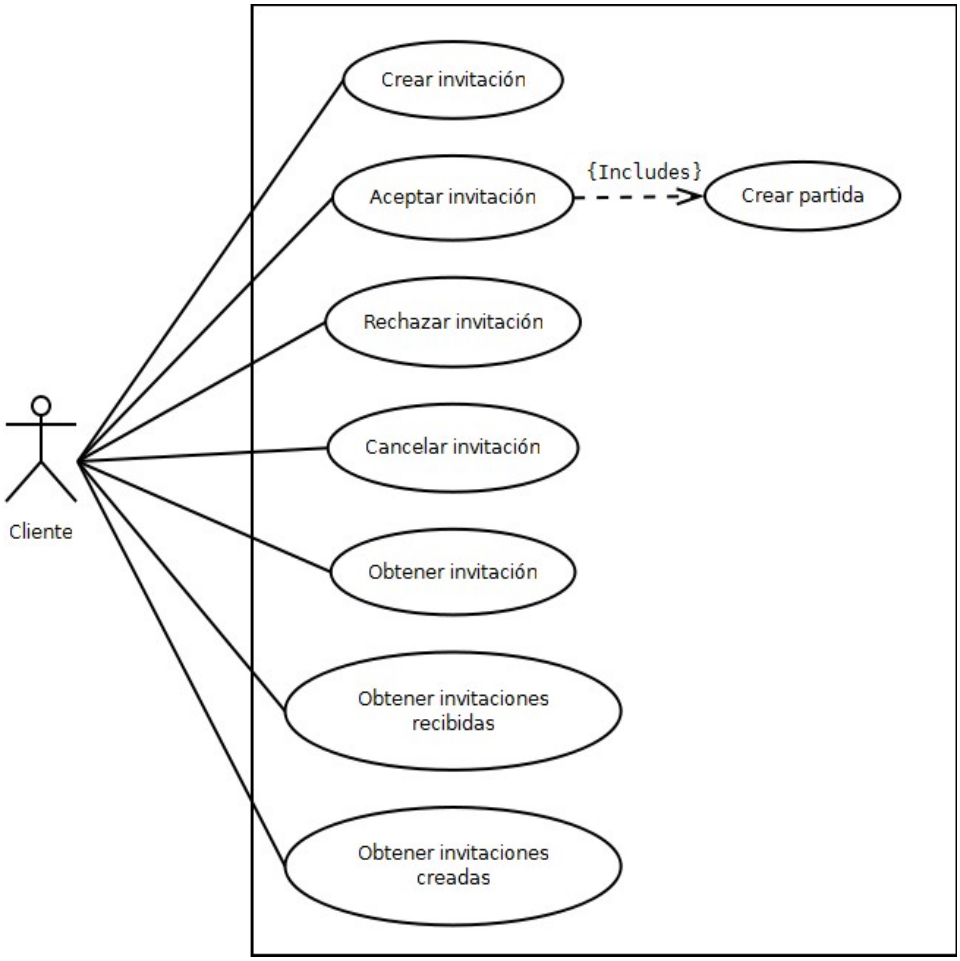

Figura 4: Casos de uso para invitaciones.

En caso de que se desee jugar contra algún desconocido, se hará una invitación general, que denominaremos petición o búsqueda de rival, en la que se establecen los requisitos que debe cumplir otro jugador para poder aceptarla. Estos requisitos se establecen en base a la puntuación Elo del creador de la petición y quien desee aceptarla. Al igual que en las invitaciones, un jugador puede consultar qué peticiones ha creado y cancelarlas, si lo considera necesario. Todo jugador podrá obtener una lista de todas las peticiones que podría aceptar, teniendo también la opción de restringir las peticiones en función del Elo del creador.

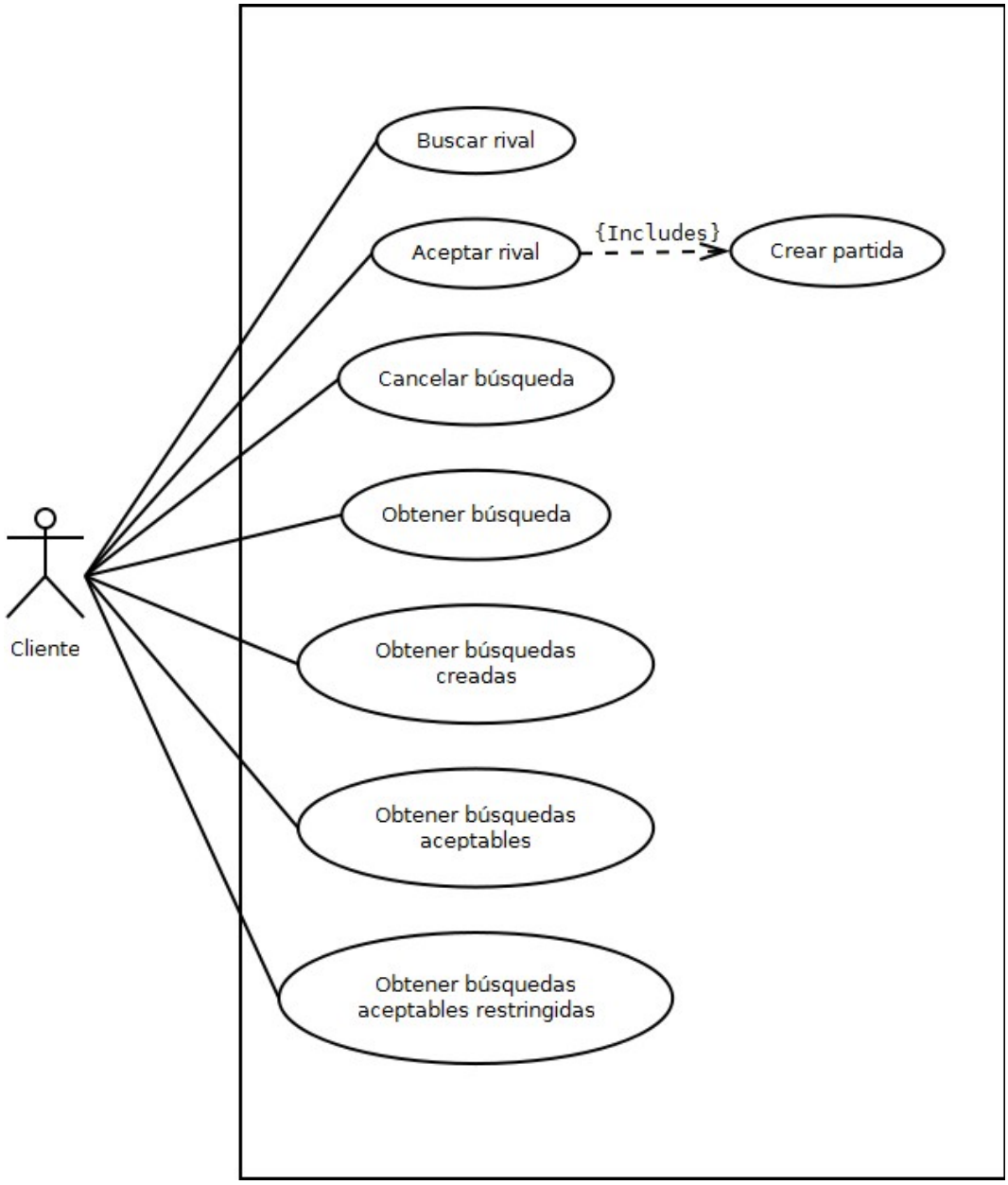

Figura 5: Casos de uso para peticiones de rival.

4.1.3 Gestión de partidas

Los jugadores podrán consultar las partidas en que han participado, están jugando o en las que les toca mover. Es además posible pedir partidas en particular, basándose en su identificador, lo que permite seguir partidas de terceros.

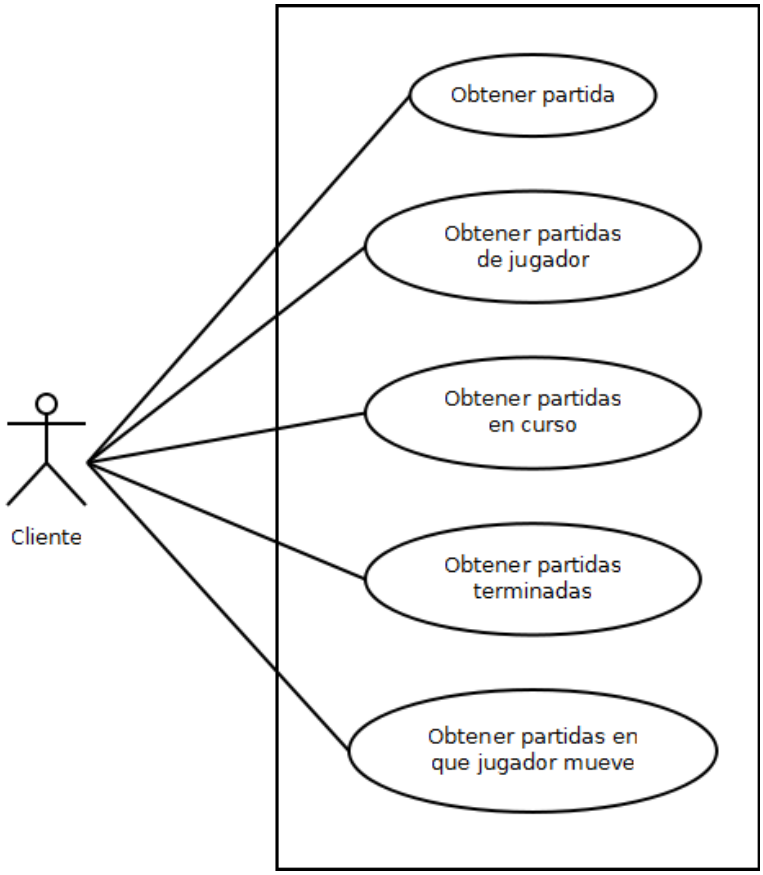

Figura 6: Casos de uso para partidas.

4.1.4 Jugadas

En las partidas en que un jugador esté participando y le corresponda el turno, éste tendrá la opción de ejecutar movimientos válidos, entre los que se cuenta la rendición, solicitar tablas por reglas y por acuerdo. En caso de haber recibido una petición de tablas, un jugador podrá aceptarlas o rechazarlas.

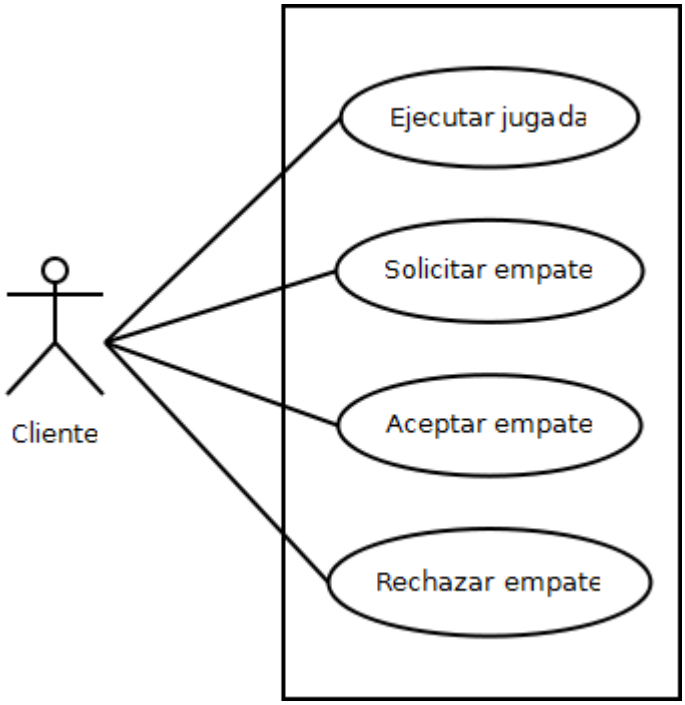

Figura 7: Casos de uso para jugadas.

4.2 Requisitos

Los casos de uso presentados anteriormente pueden usarse para determinar los requisitos que debe cumplir el sistema a desarrollar. Éstos se dividirán en requisitos de usuario (RU) y de arquitectura (RA).

4.2.1 Requisitos de usuario

RU001 – Identificación de usuarios
Un usuario debe identificarse antes de poder realizar otras operaciones en el sistema.

Tabla 1: RU001 - Identificación de usuarios.

RU002 – Métodos de identificación
Para identificarse un usuario puede proveer el nombre y clave de una cuenta existente o crear una nueva.

Tabla 2: RU002 - Métodos de identificación.

RU003 – Eliminación de cuentas
Debe permitirse eliminar una cuenta del sistema sin afectar al resto de funcionalidad (puntuación Elo, histórico de partidas...).

Tabla 3: RU003 - Eliminación de cuentas.

RU004 – Inicio de partidas
Las partidas sólo se iniciarán cuando se acepte una invitación o una petición de rival.

Tabla 4: RU004 - Inicio de partidas.

RU005 – Consulta de invitaciones y peticiones
Las invitaciones o peticiones de rival deben poder ser consultadas por sus creadores y los posibles receptores.

Tabla 5: RU005 - Consulta de invitaciones y peticiones.

RU006 – Eliminación de invitaciones y peticiones
Las invitaciones y peticiones de rival pueden ser eliminadas por sus creadores.

Tabla 6: RU006 - Eliminación de invitaciones y peticiones.

RU007 – Gestión de invitaciones
Las invitaciones pueden ser aceptadas o rechazadas por sus receptores. Ambas acciones suponen la eliminación de la invitación.

Tabla 7: RU007 - Gestión de invitaciones.

RU008 – Gestión de peticiones de rival
Las peticiones de rival pueden ser aceptadas por cualquier jugador que cumpla las condiciones establecidas. Aceptar una petición de rival supone la eliminación de la misma.

Tabla 8: RU008 - Gestión de peticiones de rival.

RU009 – Consulta de partidas
Cualquier jugador puede consultar cualquier partida, en curso o finalizada, sea partícipe o no de ella.

Tabla 9: RU009 - Consulta de partidas.

RU010 – Ejecución de jugadas
Un jugador sólo podrá mover en aquéllas partidas no terminadas en que participe y sea su turno.

Tabla 10: RU010 - Ejecución de jugadas.

RU011 – Fin de partida
Una partida terminará cuando uno de los jugadores ejecute un jaque-mate, se rinda, no ejecute un movimiento en el tiempo estipulado o se acepte una petición de tablas por reglas o acuerdo.

Tabla 11: RU011 - Fin de partida.

RU012 – Tablas por reglas
Las tablas por reglas serán aceptadas si la partida en el turno del jugador cumple las normas de triple repetición o de los 50 movimientos, o si el movimiento con que el jugador realiza la petición dejaría la partida en una situación en que alguna de ellas se cumpliese.

Tabla 12: RU012 - Tablas por reglas.

RU013 – Tablas por acuerdo
Cuando un jugador solicite tablas se pasará el turno parcialmente al oponente, sólo pudiendo éste aceptar o rechazar las tablas. Si las rechaza, el turno volverá al jugador que las solicitó.

Tabla 13: RU013 - Tablas por acuerdo.

RU014 – Puntuación Elo
Se mantendrá la información de la puntuación Elo de todos los jugadores, actualizándola al final de cada partida jugada.

Tabla 14: RU014 - Puntuación Elo.

4.2.2 Requisitos de arquitectura

RA001 – Aislamiento de la base de datos
Los clientes no se comunicarán directamente con la base de datos, para evitar ataques mediante consultas preparadas.

Tabla 15: RA001 - Aislamiento de la base de datos.

RA002 – Limitación de caracteres en cuentas y claves.
No todos los caracteres serán aceptables en nombres de usuarios y claves, para evitar ataques mediante consultas preparadas.

Tabla 16: RA002 - Limitación de caracteres en cuentas y claves.

RA003 – Acceso multiplataforma del servidor
El servidor debe ofrecer funcionalidad para ser usado tanto a través de terminales .NET como de otras plataformas, a través de SOAP u otros mecanismos de comunicación independientes de plataforma.

Tabla 17: RA003 - Acceso multiplataforma al servidor.

RA004 – Acceso múltiple concurrente
El servidor debe permitir a varios usuarios conectarse y realizar peticiones a la vez, sin necesidad de reiniciar sesiones.

Tabla 18: RA004 - Acceso múltiple concurrente.

4.3 Diseño

A continuación presentaremos los diagramas de clases de cada uno de los componentes de ChessAsin.

4.3.1 ServerChessDatabase

A la derecha en la Figura 8 se encuentran las clases de envoltura para datos de la base de datos. Cada uno de los Manager, implementado como un *Singleton*, se encarga de gestionar el acceso a cada tabla de la base de datos, y son accedidos desde *ChessServerDB*. Los dos tipos enumerados, *ColourChoice* y *RivalRating*, se usan para mapear datos enumerados de la base de datos, por lo que sus posibles valores deben corresponderse exactamente con los aceptados en las tablas.

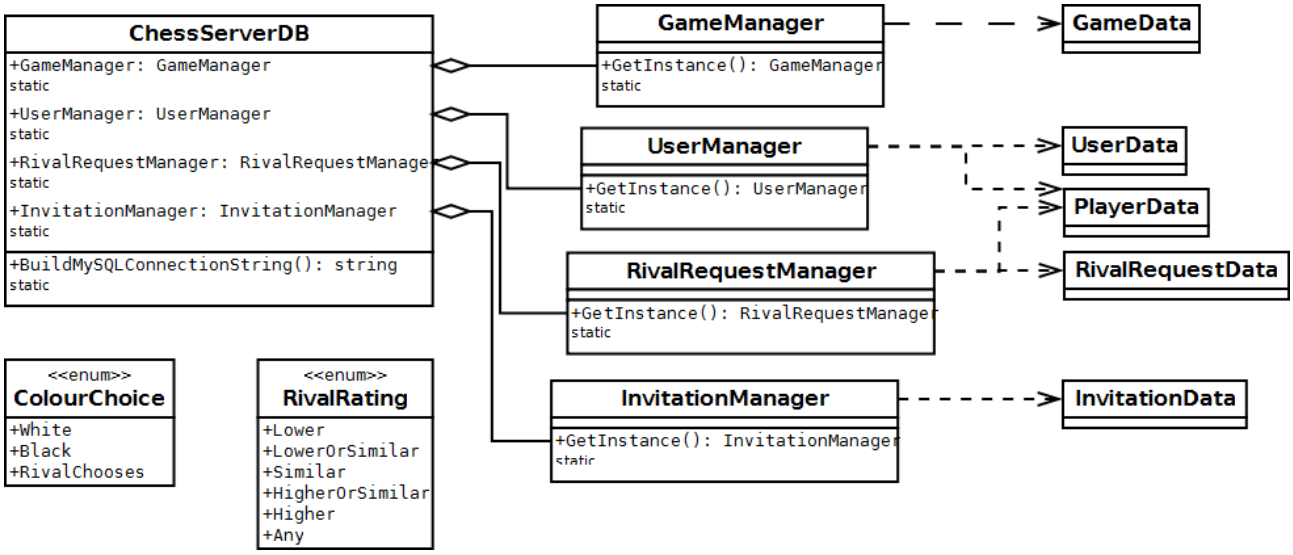

Figura 8: Diagrama de clases de ServerChessDatabase.

4.3.2 ChessServiceLibrary

En la Figura 9 se listan todos los miembros de contrato definidos, que permitirán el envío de estructuras complejas entre el servidor y los clientes. Se han omitido las clases de envoltura de la base de datos, aunque la biblioteca se encarga de la conversión de éstas a los contratos correspondientes. Tampoco se muestra la lista completa de métodos que *IChessService* define, para mantener un gráfico legible.

CollectionContract se utiliza para devolver a los clientes varios contratos en en una única respuesta, como pueden ser varios *GameContract* o *InvitationContract*. Por su parte, *ReplyContract* permite encapsular cualquier contrato de respuesta en una estructura que indica si la petición del cliente fue aceptada o rechazada, añadiendo mensajes informativos en el segundo caso.

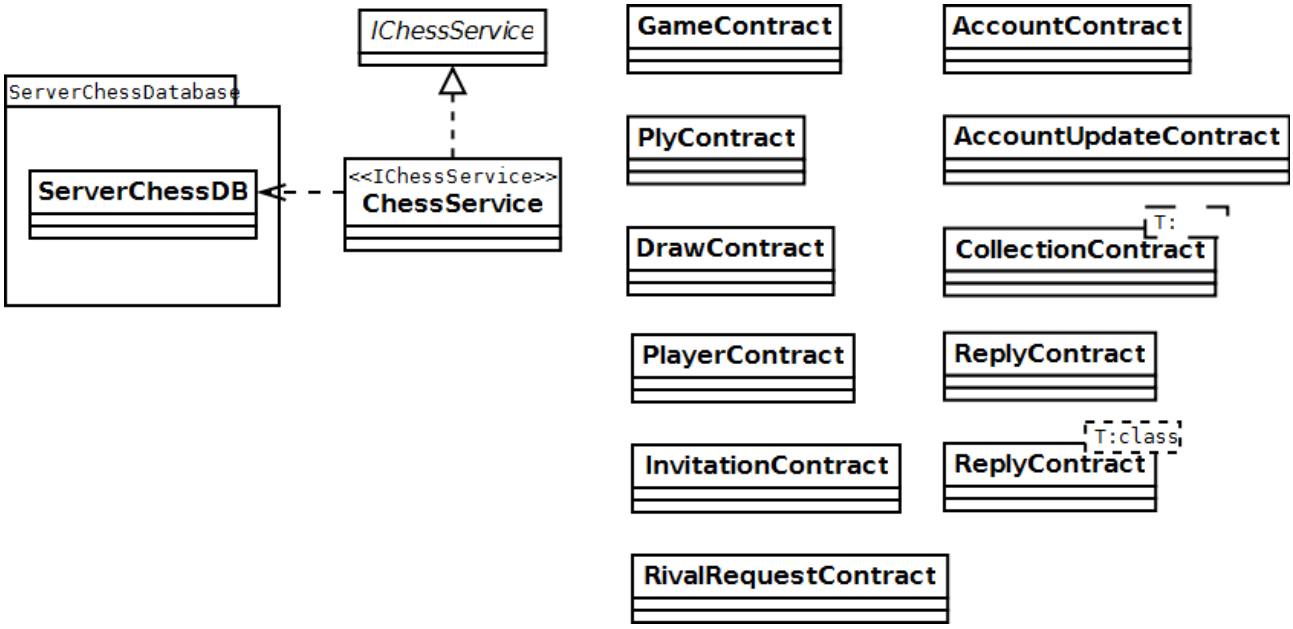

Figura 9: Diagrama de clases de ChessServiceLibrary.

4.3.3 ChessAsin

El motor de ajedrez implementado para el proyecto se encuentra estructurado como se muestra en la Figura 10.

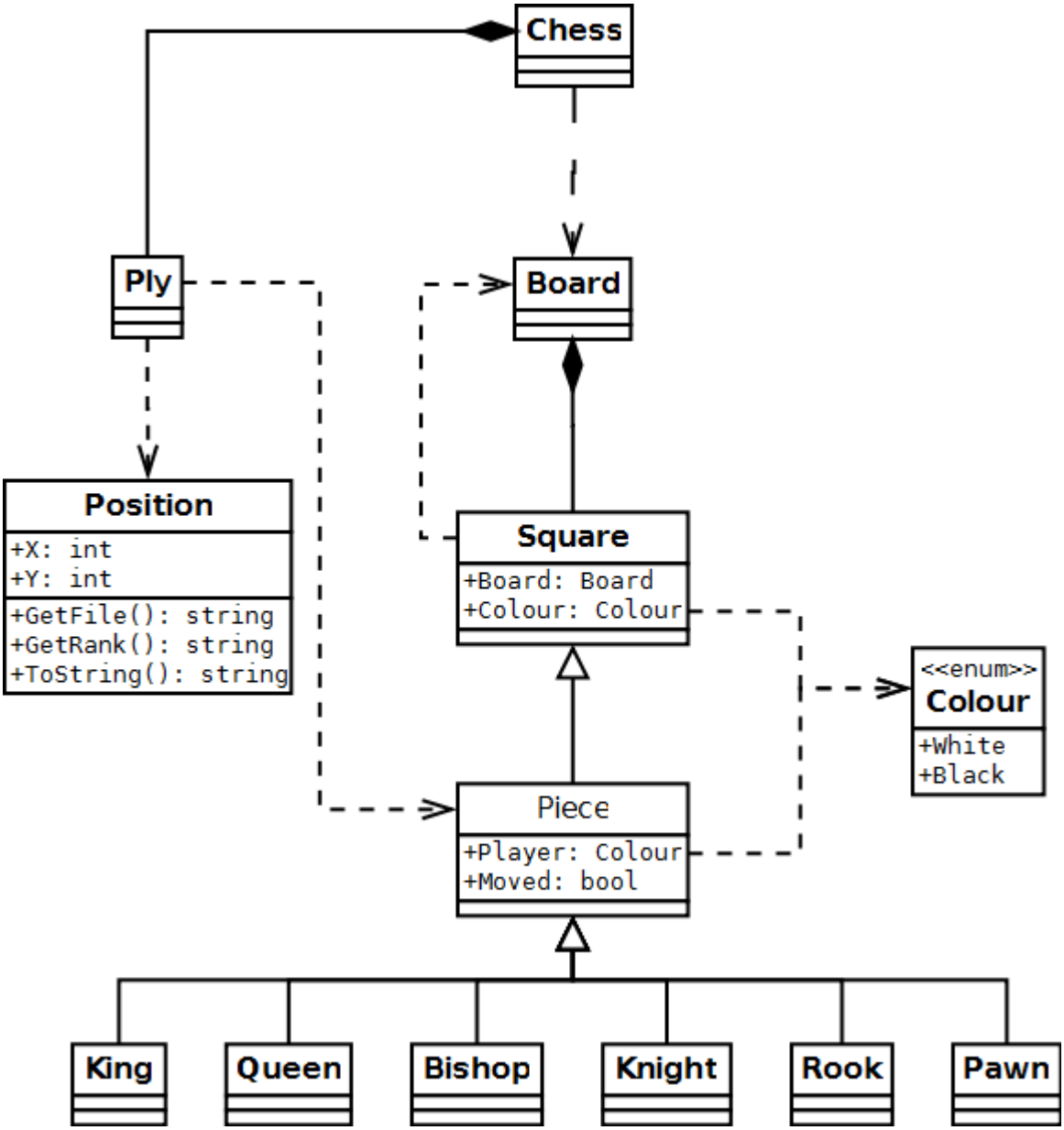

Figura 10: Diagrama de clases de ChessAsin.

4.3.4 Cliente

El cliente, por haber requerido la implementación de componentes específicos, es más complejo que los proyectos anteriores. Para facilitar la comprensión, lo dividiremos en varios diagramas de clases, en función de los paquetes de que se compone.

Empezaremos con la jerarquía de componentes de interfaz.

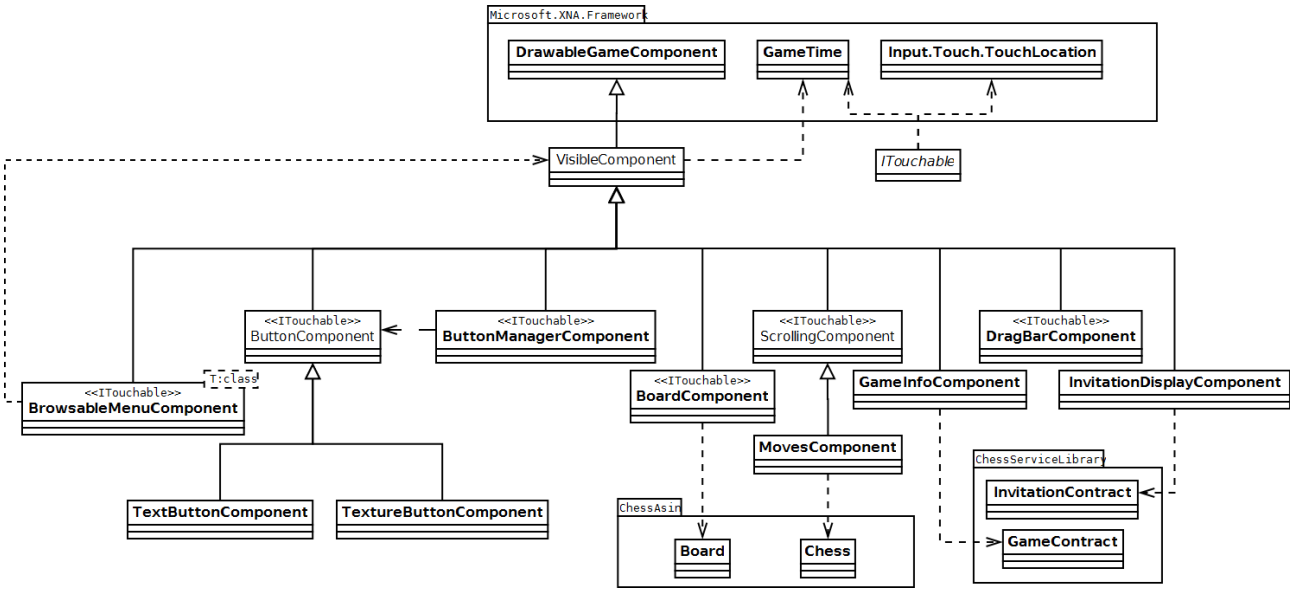

Figura 11: Diagrama de clases de VisibleComponent y ITouchable.

Y la general del juego. *Scene* es la clase base de la jerarquía que define cada una de las pantallas que se presentan al usuario. En todo momento se mantiene en *GlobalInfo* una referencia a la escena actualmente en pantalla, más aquéllas de las que se quiera mantener una única copia. Por ser la versión de Windows del cliente se incluye *MouseManager*, que simula la interfaz táctil de Windows Phone 7 con el ratón.

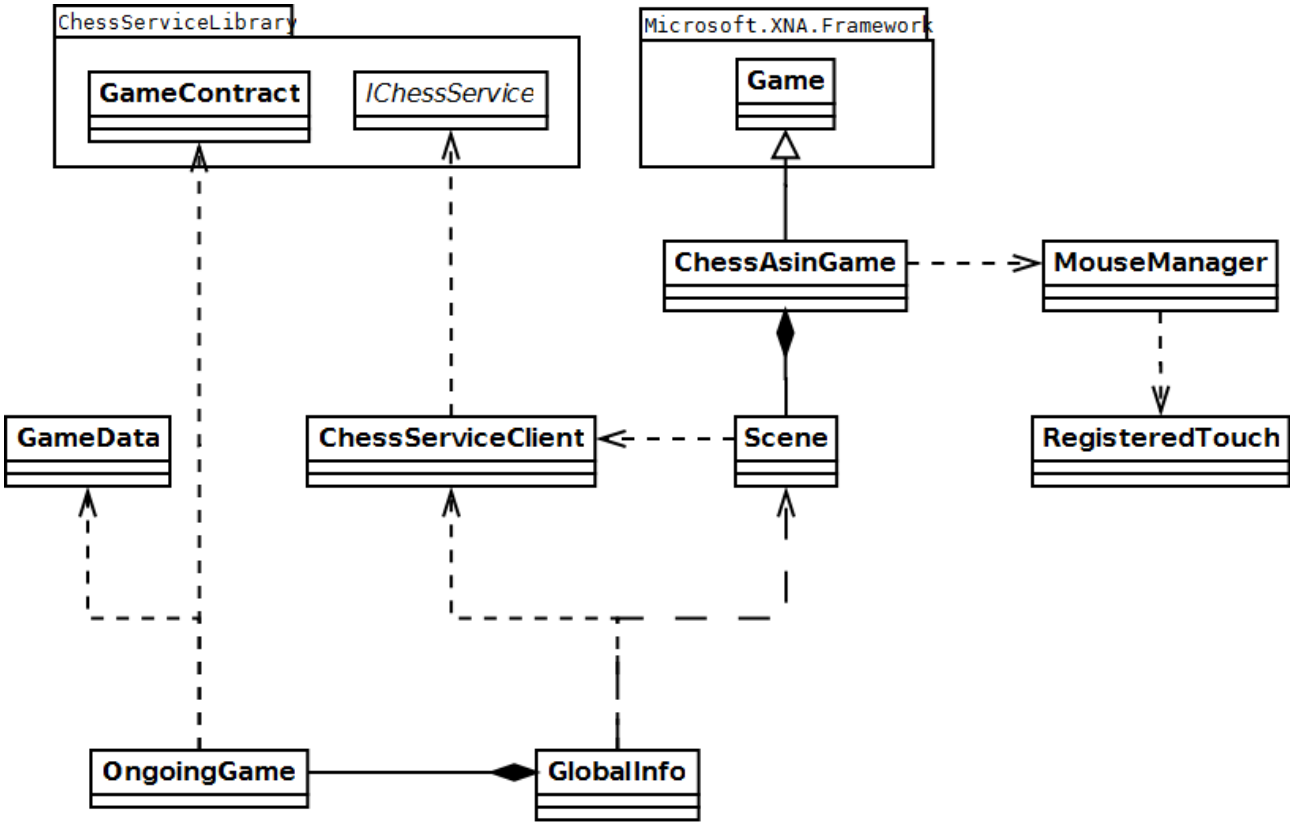

Figura 12: Diagrama de clases de WinChess.

4.4 Implementación

A continuación explicaremos, sin entrar en excesivo detalle, cómo se ha procedido a implementar cada componente del sistema, incluyendo el cliente. Empezaremos con una breve introducción a las principales tecnologías utilizadas en el desarrollo, para después analizar cada componente.

4.4.1 Tecnologías

Como primer paso, haremos un repaso de las diversas tecnologías que se han utilizado en la implementación del proyecto.

4.4.1.1 Subversion

Subversion es un sistema de control de revisiones de código abierto, mantenido por Tigris. Utilizándolo se puede mantener un histórico de los cambios realizados en el proyecto, facilitando tareas complejas como deshacer cambios que introdujeran errores, introducir modificaciones

experimentales sin afectar al desarrollo principal del proyecto o integrar estos experimentos con la rama principal.

Con el fin de mantener una copia de seguridad y control de versiones, el proyecto se encuentra en un servidor público de Subversion mantenido por la empresa Assembla. En éste, con cuentas a las que se concedan los permisos adecuados, se pueden encontrar todas las partes necesarias para recrear el servidor y cliente:

- Código de todos los componentes del servidor y de los clientes, junto con los archivos de cada solución.
- Proyecto de MySQL Workbench con el que se diseña y gestiona la base de datos.
- Scripts de creación de las tablas de la base de datos.
- Configuraciones del servidor de ChessAsin.

4.4.1.2 MySQL

MySQL es un sistema de gestión de bases de datos de código abierto, actualmente perteneciente a Oracle. MySQL es una de las soluciones más populares, dado su bajo coste, buena documentación, la posibilidad de contratar servicios de asistencia técnica y su amplia funcionalidad.

Existen además diversas herramientas diseñadas para facilitar la tarea de gestionar bases de datos MySQL. Para ChessAsin se ha utilizado MySQL Workbench, que incluye todo lo necesario para llevar una base de datos desde la fase de diseño a la de mantenimiento.

Originalmente se pretendía instalar una base de datos Microsoft SQL Server, para aprovechar las facilidades que .NET ofrece para colaborar con dicho sistema. En particular se pretendía explotar LINQ to SQL, que habría permitido implementar toda la biblioteca de acceso a la base de datos sin una sólo línea de SQL. Finalmente, la excesiva complejidad del proceso de instalación, así como la falta de herramientas de gestión que cumplieran nuestras expectativas, hizo que se abandonara esta opción.

A pesar de la mayor complejidad del código necesario para interactuar con la base de datos, MySQL ofrece compensaciones por medio de sus potentes herramientas de gestión y la posibilidad de mover más adelante el sistema a plataformas distintas, como Linux, sin excesivos problemas.

4.4.1.3 Windows Communication Foundation

Biblioteca de comunicación por red, con especial énfasis en servicios de red. Define su propio esquema de comunicación, basado en las capacidades de serialización de .NET, pero permite habilitar la compatibilidad con aplicaciones de otras plataformas, generando y aceptando mensajes codificados según el estándar SOAP.

WCF facilita la gestión de sesiones, pudiendo el programador delegar toda la lógica relacionada con las mismas a WCF. Esto incluye la cancelación de sesiones en caso de errores de comunicación (identificación fallida), la obligación de iniciar toda comunicación con peticiones determinadas (login) y cerrarlas automáticamente tras la recepción de mensajes especificados por el programador (logout).

La única limitación impuesta por WCF es que el servicio a exponer debe encontrarse en una única interfaz, que será implementada por la clase que responda desde el servidor. Esto hace que las clases de servicio sean de un tamaño considerable, aunque esto es salvable usando clases parciales.

4.4.1.4 XNA 4.0

Biblioteca de implementación de juegos para Windows, Xbox 360 y Windows Phone 7, desarrollada por Microsoft sobre la plataforma .NET. Su diseño se basa en una arquitectura de componentes jerárquicos, que al ser registrados en el sistema son invocados de forma automática. Aunque esta aproximación resulta muy cómoda para proyectos pequeños, en casos más complicados suele buscarse una forma de mantener más control en el lado del programador, añadiendo algún sistema de discriminación entre objetos a pintar, actualizar o interactuar.

4.4.2 Base de datos

La base de datos ha sido creada utilizando la plataforma MySQL, en su versión GPL. MySQL es uno de los líderes mundiales en sistemas de gestión de bases de datos (SGBD), por ser gratuita, robusta y contar con gran cantidad de aplicaciones para facilitar su instalación y mantenimiento.

El diseño de la base de datos se muestra en el esquema entidad-relación extendido de la Figura 13.

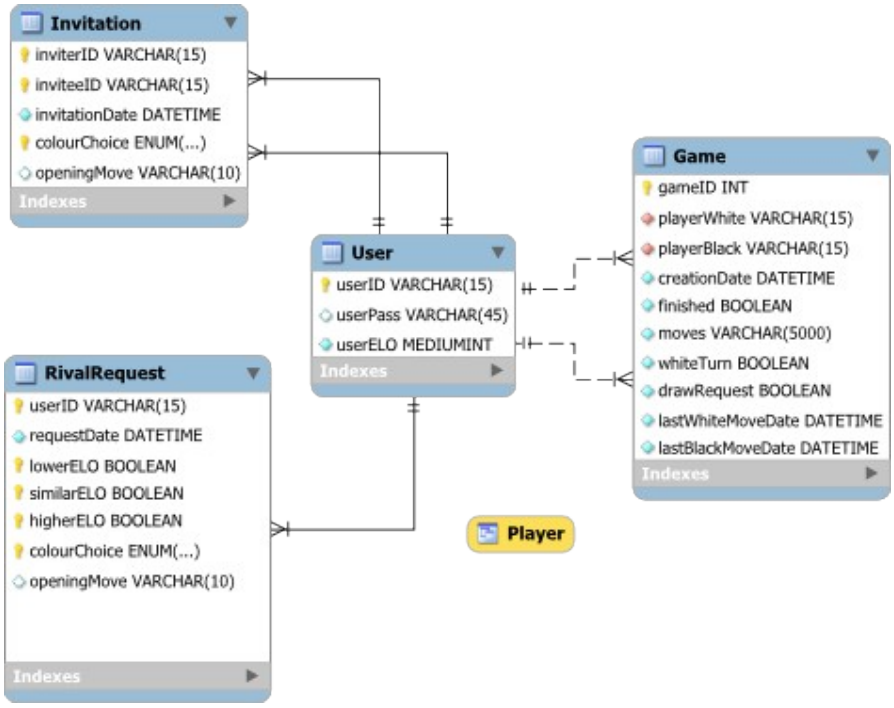

Figura 13: Diagrama Entidad-Relación extendido.

Este diagrama fue creado con MySQL Workbench, haciendo uso de las extensiones que ofrece sobre el modelo Entidad-Relación básico. Workbench facilita el paso del esquema a código SQL, haciendo que sólo sea necesario revisar el resultado y realizar las adaptaciones precisas en formatos de tabla, fechas y similares.

Además de los datos mostrados en el esquema, la base de datos contendrá los procesos almacenados que se consideren necesarios. Éstos se limitarán a operaciones comunes que requieran una gran cantidad de accesos a la base de datos, con el fin de reducir la cantidad de peticiones externas.

Para facilitar el uso en entornos internacionales, las fechas se almacenarán según el formato UTC (*Coordinated Universal Time*), dejando la conversión a tiempo local en manos de los clientes.

4.4.2.1 Mecanismo de acceso

El acceso a la base de datos se hace a través de una conexión TCP/IP local, usando una cuenta de privilegios reducidos cuyos datos serán eventualmente movidos a un archivo de configuración. Como medida de seguridad, la base de datos no expone ningún puerto de comunicación fuera de la máquina en que se ejecuta, puesto que sólo la biblioteca de acceso necesita conectarse a ella.

4.4.3 ServerChessDatabase

Ésta es la biblioteca encargada de hacer de intermediaria entre la base de datos y el propio servidor de ChessAsin. Su existencia permite mantener toda la lógica de la base de datos aislada del servidor, facilitando el mantenimiento, evaluación y prueba.

El acceso a la base de datos se hace a través de clases específicas, gestionadas mediante *singletons*, cada una de ella controlando el acceso a diferentes tablas de la base de datos. En total hay cuatro clases gestoras, llamadas *UserManager*, *GameManager*, *InvitationManager* y *RivalRequestManager*.

También se ofrece en esta biblioteca funcionalidad para convertir formatos de fechas entre el usado por la base de datos y el sistema, aparte de otras facilidades para abrir conexiones con la base de datos.

4.4.3.1 Clases de envoltura

Para poder tratar los datos contenidos en la base de datos desde C# es necesario aplicar diversas transformaciones a los mismos, por no ser completamente equivalentes. En *ServerChessDatabase* se han incluido varias clases de envoltura, preparadas para contener la información relativa a cada tipo de consulta soportada por la biblioteca.

Éstas clases no se limitan a encapsular toda la información de cada tabla, pues muchas veces no todos los datos son necesarios o deseables. Por ejemplo, *PlayerData* contiene los datos de un jugador, omitiendo la clave de su cuenta.

4.4.3.2 Pruebas unitarias

Se ha creado una colección de pruebas, implementadas usando NUnit, para todas las funciones expuestas por ChessServerDatabase. Se prueban tanto casos extremos como situaciones normales, para asegurar que el comportamiento de la biblioteca responde a lo esperado. En el futuro toda nueva funcionalidad deberá contar con sus propias pruebas unitarias, y los cambios en la especificación deberán evitar romper las pruebas ya creadas, o actualizarlas en caso de que la ruptura sea esperada.

Estas pruebas pueden encontrarse en el proyecto *SeverChessDBTest*.

4.4.4 ChessServiceLibrary

Ésta es la biblioteca que expone la funcionalidad del servidor ChessAsin a los clientes. Está implementada usando WCF, lo que implica la creación de una interfaz de comunicación, *IChessService*, que será la que esperen los clientes, así como una clase que implemente dicha interfaz, *ChessService*.

Para responder a las peticiones, el servicio consulta y actualiza la base de datos usando ServerChessDatabase, lo que nos permite mantener este módulo libre de lógica de datos.

Para evaluar los movimientos y demás peticiones de partidas de los jugadores se utiliza el motor de ajedrez creado específicamente para este proyecto, también llamado ChessAsin, cuya descripción se encuentra más adelante. Cabe recalcar que la base de datos no realiza ningún tipo de validación de la información que almacena, dando por supuesto que todo lo que recibe es válido.

4.4.4.1 Características de WCF

Los métodos de acceso al servicio de la interfaz y la información a intercambiar mediante ellos deben ser identificados como contratos y miembros de contrato, respectivamente. WCF se encarga de serializar cada petición y deserializar las respuestas, de acuerdo con la especificación del contrato, evitando el tener que generar e interpretar los mensajes en nuestro lado.

En caso de errores de comunicación o de que un usuario no se identifique adecuadamente, WCF se encarga de gestionar el cierre de la conexión y de informar al cliente del error, según la excepción creada por nuestro código.

Además, gracias a la configuración del servidor, éste crea un hilo de ejecución independiente para cada petición recibida, lo que nos permite mantener la información de cada conexión aislada y controlada sin estructuras o mecanismos complejos de control de acceso. Por desgracia, esta funcionalidad no se puede aprovechar desde terminales de Windows Phone 7, por ser éste incompatible con el sistema de conexión que permite gestión automática de sesiones. Para estos terminales sería preciso implementar un sistema de *binding* que cumpla las características requeridas, lo que se ha considerado que se encuentra fuera de los objetivos de este proyecto.

4.4.4.2 Pruebas unitarias

Las pruebas de este módulo se han creado mediante historias de usuario, simulando interacciones

completas y complejas de usuarios con el sistema. Se ha intentado darles la misma profundidad que con pruebas unitarias, añadiendo además comprobaciones sobre la capacidad del sistema para responder a varios usuarios de forma simultánea.

4.4.5 ChessAsin, motor de ajedrez

El motor de ajedrez creado para este proyecto permite jugar una partida de ajedrez, de acuerdo a las reglas clásicas, aunque no controla los tiempos de juego. Esta tarea se deja en manos de la interfaz o aplicación que se monte sobre ChessAsin.

Incluye, además, funcionalidad para exportar e importar partidas en notación algebraica. Carece de interfaz, por ser de naturaleza puramente lógica.

En ChessAsin el tablero se representa como una matriz de casillas de colores alternos. Las piezas son clases derivadas de casilla, por lo que sólo se necesita una consulta simple para saber si una casilla está ocupada, y por quién. Para facilitar esta tarea se aprovechan las facilidades que C# ofrece para identificar y convertir miembros de una jerarquía.

Una partida se representa con el tablero, en la configuración que corresponda, más la serie de movimientos que han llevado a ella. Los movimientos se almacenan como tuplas de jugadas de los jugadores blanco y negro.

Algunos aspectos de la implementación del ajedrez, a pesar de ser relativamente simples para un humano, presentan complejidades de consideración a la hora de implementarlas. Los más significativos son comentados a continuación.

4.4.5.1 Evaluación de movimientos válidos

Antes de permitir ejecutar un movimiento debe comprobarse si es válido para la pieza y si no deja al rey del jugador que mueva al descubierto. Para el primer caso, cada pieza, dada una posición y un tablero, puede producir la lista de casillas a las que podría desplazarse. Superado el primer paso, el movimiento se ejecuta en un tablero auxiliar, copiado del actual, y en el se comprueba si alguna de las piezas del rival amenaza la posición del rey.

La segunda comprobación no la ejecutan las piezas directamente, pues esto provoca un ciclo infinito de comprobaciones.

Un caso similar se da con la detección de jaques y mates, pues requieren buscar un movimiento del

contrario que ponga al rey a salvo de un ataque directo. La detección de ahogamiento es también parecida, requiriendo la búsqueda de un movimiento válido que no deje al rey en jaque. La resolución de estos casos es parecida a la de movimientos válidos.

4.4.5.2 Coronación

La posibilidad de coronar hace que las jugadas puedan necesitar interacción del usuario tras indicarse la dirección de destino. Esto obliga a realizar los movimientos en dos fases, con la primera obteniendo el movimiento y, si no hay coronación, invocando directamente a la segunda, que evalúa y almacena la jugada. En caso de requerir coronación, se devuelve el control a la aplicación que esté usando el motor, por medio de una función *callback* que pida la pieza a coronar al usuario. Tras determinar la pieza a que se desea coronar, la aplicación deberá invocar a la segunda fase, con los nuevos datos. Tras esto es necesario volver a evaluar el resultado de la jugada, pues es posible que se haya generado un jaque o mate.

4.4.5.3 Tablas por repetición triple o 50 movimientos

En cualquier momento de la partida se permite a un jugador solicitar tablas unilateralmente, bien con el estado actual del tablero o facilitando una jugada que las provoque. Puesto que carecemos de un sistema que nos permita comprender las intenciones del jugador, debemos evaluar los dos casos posibles.

El primero de los casos, la regla de los 50 movimientos, requiere comprobar si se han ejecutado 50 movimientos (50 jugadas por jugador) o más sin haberse desplazado ningún peón o producirse una captura. El segundo, si a lo largo de la partida se ha dado exactamente la disposición actual de piezas en el tablero un mínimo de 3 veces, con el mismo jugador moviendo y las opciones de enrocar y capturar *en passant* idénticas para ambos jugadores.

En caso de que el jugador dé el movimiento con que desea pedir tablas, se ejecuta dicho movimiento en una nueva instancia de *Chess*, copia del juego, y se usa ésta como referencia.

Para evaluar tanto los 50 movimientos como la repetición triple se ha implementado un método, *Chess.IsMovesDrawValid*, que crea un tablero auxiliar en el que ejecutan todos los movimientos de la partida hasta el presente. Durante la ejecución se cuenta cuántas veces se ha producido una configuración igual a la actual, incluyendo la disposición inicial del tablero. Además, se incrementa un contador cada vez que una jugada no es de peón o no produce una captura, o se devuelve a 0 en

caso contrario. Si tras la ejecución de los movimientos el contador de repetición es mayor que 3 o el de 50 movimientos mayor de 100, se acepta la petición de tablas y la partida concluye.

Dado lo costoso del proceso, esta evaluación sólo se realiza bajo petición. Y, aunque es cierto que podría mantenerse la cuenta de movimientos sin captura o peón movido, no se consideró que valiese la pena, dada la escasa frecuencia con que éstas peticiones se deberían producir.

4.4.6 WinChess, cliente para ChessAsin

WinChess es la aplicación cliente del sistema, creada para probar su funcionamiento. Utiliza ChessAsin como motor de ajedrez y se comunica con el servidor mediante WCF. Ha sido implementado usando el motor para juegos XNA, de Microsoft, y, a pesar de estar diseñado con Windows como plataforma de destino, se ha creado asegurando que sea portable a Windows Phone 7. Para ello ha sido necesario crear una clase de envoltura que simule el comportamiento de una interfaz táctil compatible con la de WP7.

Una vez se resuelvan las limitaciones de WCF que impiden ejecutar el cliente en WP7, la conversión del cliente a la plataforma móvil debería requerir solamente añadir los mecanismos apropiados para almacenar información en el móvil, controlar los estados de reposo de la aplicación y las notificaciones del usuario.

4.4.6.1 Interfaz de usuario

Debido a que XNA carece de las herramientas y componentes necesarios para crear una interfaz, éstos tuvieron que ser implementados en nuestro lado de la aplicación. Para ello, y siguiendo el principio de diseño basado en componentes de XNA, se ha creado una jerarquía de componentes visibles, a partir de nuestro *VisibleComponent*, que hereda del *DrawableGameComponent* de XNA. De esta forma podemos controlar qué componentes se dibujan en cada momento, sin tener que preocuparnos de los componentes registrados en el sistema. Para determinar qué *VisibleComponent*s deben dibujarse en pantalla, se utiliza una jerarquía de escenas, cada una con sus propios componentes y comportamiento. Si se desea cambiar la pantalla, basta con cambiar la escena actual a una instancia de la deseada.

El mismo principio se ha utilizado para gestionar la entrada de datos en el sistema. Como la plataforma de destino es un móvil con pantalla táctil, los componentes interactivos implementan la interfaz ITouchable que, una vez más, se controla desde la escena. Éstas son las encargadas de

comprobar qué componentes interactivos han sido tocados por el usuario, delegando en los componentes el determinar si les corresponde o no responder a la interacción.

4.5 Conclusiones

En este capítulo hemos presentado las tecnologías y herramientas usadas para desarrollar ChessAsin, así como los detalles de implementación de la base de datos, el servidor y el cliente.

En resumen, nuestro sistema ofrece a los jugadores la funcionalidad para crear o eliminar cuentas, interactuar con otros jugadores, estudiar partidas acabadas o en curso y, por supuesto, jugar al ajedrez. Al haber escogido WCF como base para nuestro servicio, nos aseguramos de que será posible dar soporte a toda plataforma que permita la comunicación a través de SOAP, pertenezcan o no al entorno de .NET.

Además del servidor, el componente más importante de los desarrollados es el propio motor de ajedrez, usado tanto por el servicio web como por el cliente. Es, posiblemente, el subsistema más complejo de los que se han implementado, dado que contempla las reglas completas del ajedrez, con la excepción del control de tiempo.

En los extremos de ChessAsin podemos encontrar la base de datos, donde la información de jugadores, partidas e invitaciones se almacena, y el cliente, que hace uso del servidor para acceder a esos datos. Cualquiera de los dos podría ser reemplazado en el futuro por otros basados en tecnologías distintas, pues el servidor abstrae su funcionalidad del resto de componentes.

Hemos de destacar también, aunque no sea parte integral del proyecto, la creación de los componentes de interfaz gráfica, necesarios para permitir a los usuarios interactuar con el cliente en XNA. También debemos mencionar el esfuerzo que se ha hecho por mantener la implementación del cliente dentro de las limitaciones impuestas por el .NET Compact Framework, para facilitar una futura versión compatible con Windows Phone 7.

Capítulo 5: Resultados

En este capítulo analizaremos el estado final del proyecto, una vez terminado el desarrollo, incluyendo pruebas de rendimiento y usabilidad. Nos centraremos principalmente en el servidor, aunque también trataremos, aunque en menor medida, el cliente desarrollado como prueba de concepto.

5.1 Rendimiento

A continuación detallaremos las pruebas de rendimiento que se ejecutaron, con el fin de evaluar la carga de trabajo máxima que el servidor podría llegar a soportar. Primero describiremos la metodología usada para ejecutar las pruebas, para más tarde describir las pruebas y acabar con las conclusiones derivadas de las mismas.

5.1.1 Metodología de las pruebas

Para detectar las áreas que más influencia tienen en el rendimiento general, desde el punto de vista de los clientes, primero se ejecutaron una serie de sesiones de *profiling*. Durante éstas se utilizó el cliente de forma habitual, realizando invitaciones, aceptándolas, viendo las partidas disponibles para los jugadores y jugando una serie de movimientos de dichas partidas. Con los datos obtenidos se pudo comprobar que la tarea que más costaba realizar tanto al cliente como al servidor era la recreación de partidas. En particular, el principal escollo se encuentra en el método *Chess.Parse*, usado por el servidor para reconstruir cada partida a partir de su notación algebraica cuando se solicita ejecutar un movimiento.

Una vez detectado el principal cuello de botella para el rendimiento, se ejecutaron una serie de pruebas que explotaban este punto débil de la arquitectura. El objetivo era determinar el impacto que una sobrecarga de trabajo tendría sobre los tiempos de respuesta a esperar en los clientes, por lo que se consideró razonable limitarse a probar el peor caso posible: ejecución de un gran volumen de jugadas concurrentes, en partidas de longitud considerable.

Las pruebas se realizaron en línea, con diversos ordenadores simulando hasta 8 clientes cada uno. Para mejorar los tiempos de ejecución de los clientes, se independizó cada una de las partidas a ejecutar en un hilo, permitiendo aprovechar al máximo la comunicación asíncrona con el servidor. Los participantes en cada partida se turnan para ejecutar los movimientos que les correspondiesen,

una vez el servidor confirma la jugada anterior.

La principal preocupación era evitar sobrecargar los sistemas cliente, por lo que se realizaron una serie de pruebas, determinando que el número óptimo de partidas por ordenador sería de 4.

El servidor usado en las pruebas se ha instalado en un ordenador de cuatro núcleos, ninguno de ellos virtualizado, y con 4GiB de RAM. Es de esperar que un servidor de gama profesional pueda mejorar considerablemente los resultados obtenidos en este equipo.

Debe destacarse que las pruebas que se ejecutaron son especialmente duras. Dada la naturaleza del programa, la inmensa mayoría de peticiones recibidas por el servidor en circunstancias normales no serán de ejecución de movimientos, sino operaciones menos exigentes, como peticiones de partidas en juego o de invitaciones disponibles, resueltas en reducidas consultas a la base de datos. Además, raramente se producirá una recepción masiva de peticiones al nivel que estamos probando, a no ser que el número de usuarios conectados en un momento dado se cuente en los miles.

5.1.2 Pruebas

Para establecer un caso de partida para el rendimiento del servidor primero se ejecutaron varias pruebas de una única partida de 80 movimientos (160 jugadas en total) entre dos clientes. Los tiempos de respuesta del servidor ante cada petición de movimiento se reflejan en la Figura 14.

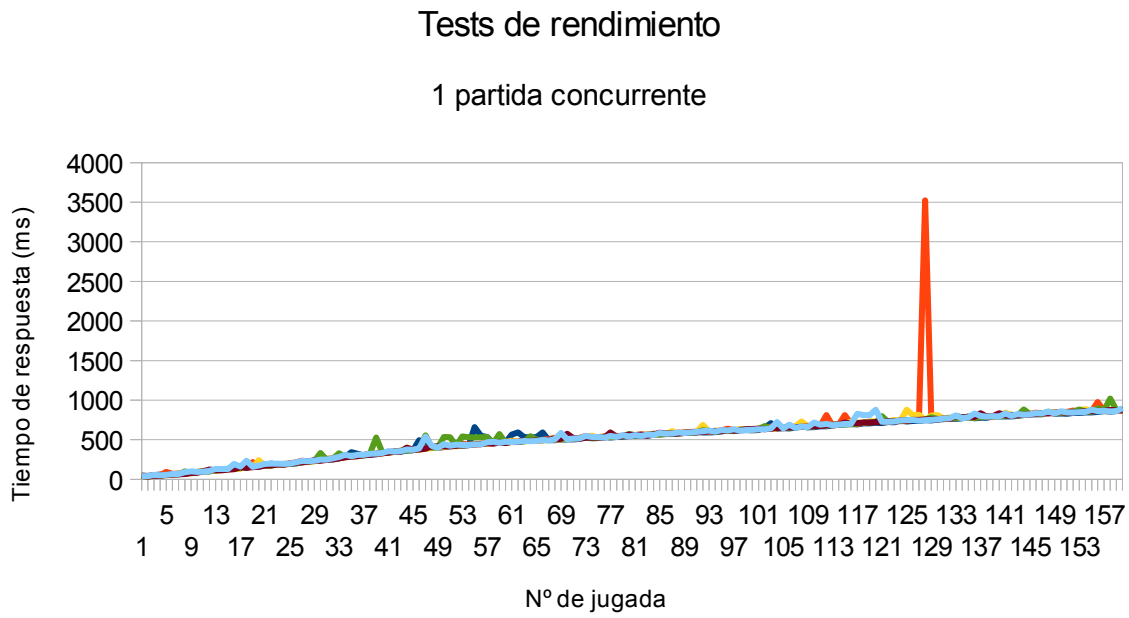

Figura 14: Tiempos de respuesta: 1 partida concurrente.

Excluyendo el dato atípico de 3,5s, se puede observar que el tiempo de respuesta aumenta en proporción casi directa con la longitud de la partida en juego, llegando casi al segundo a los 80 movimientos. Para partidas más largas el resultado empeora proporcionalmente.

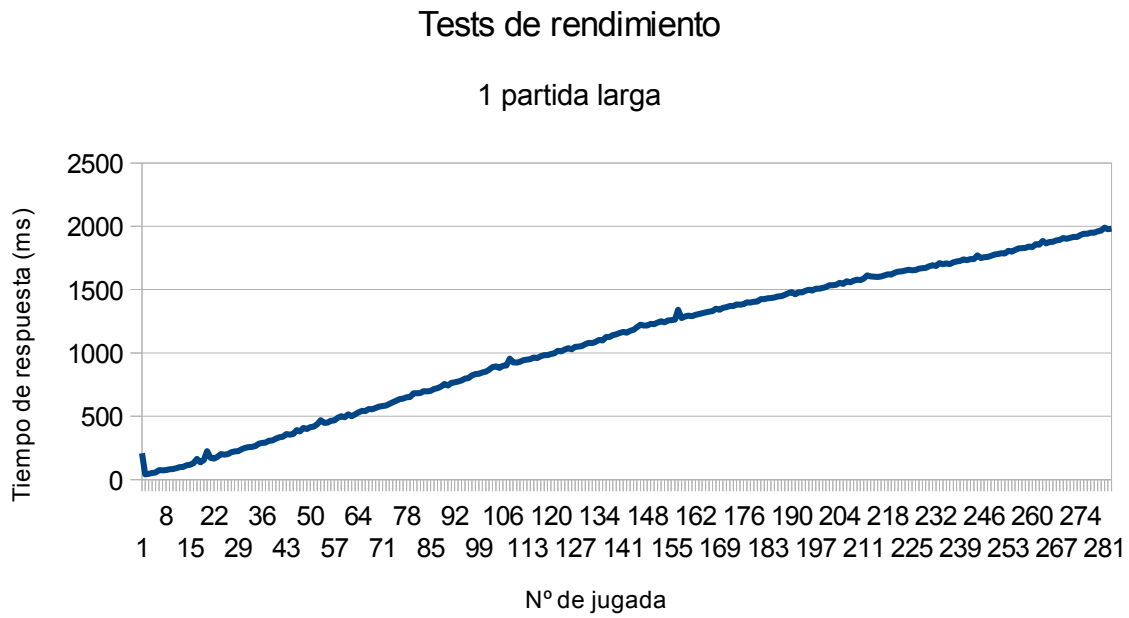

Figura 15: Tiempos de respuesta: 1 partida de 142 movimientos.

A continuación incrementamos progresivamente el número de clientes que se conectaban al servidor. Los resultados de cada prueba individual se incluyen a continuación.

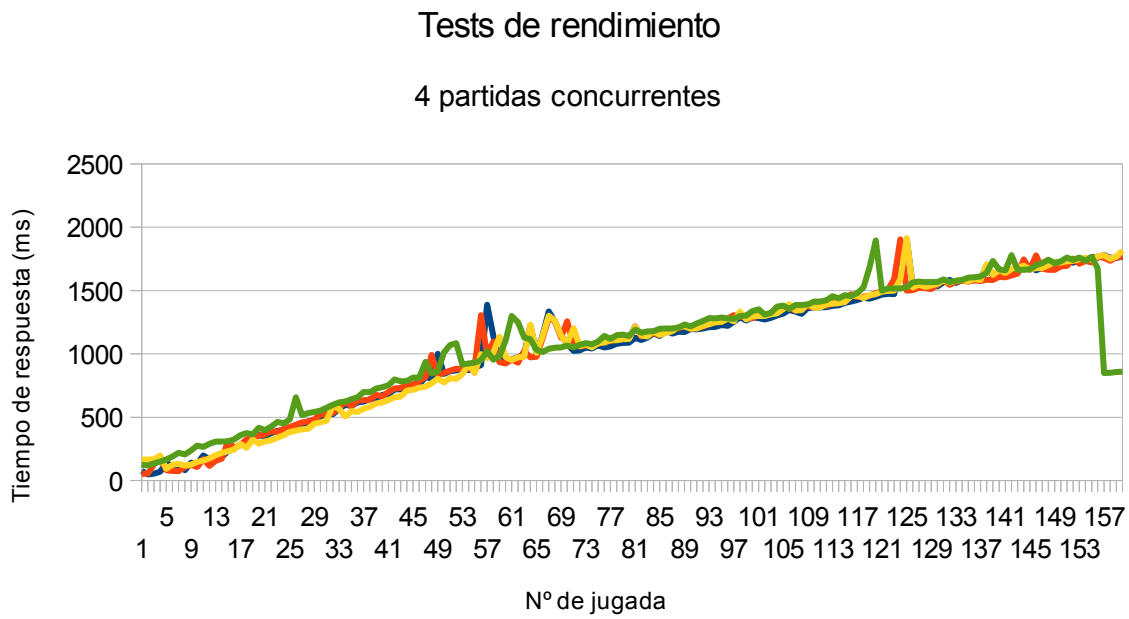

Figura 16: Tiempos de respuesta: 4 partidas concurrentes.

La carga de trabajo adicional es claramente notable, pasando el tiempo máximo a estar próximo a los 2 segundos.

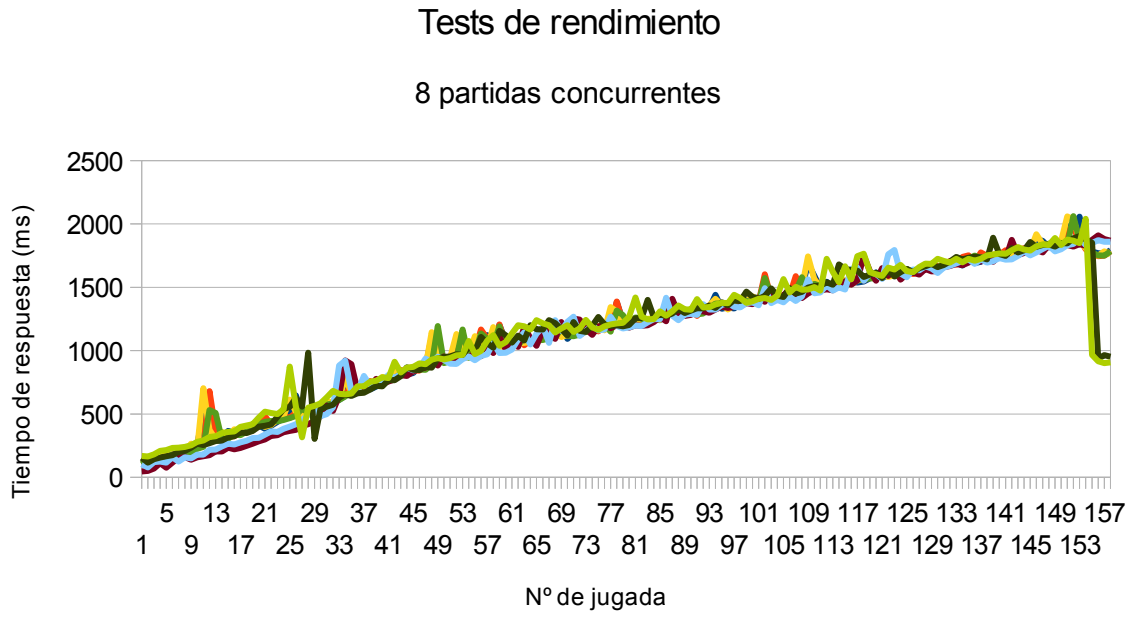

Figura 17: Tiempos de respuesta: 8 partidas concurrentes.

El impacto de incrementar el número de clientes a 16 no es proporcional al aumento de la carga, aunque sí se puede apreciar una reducción en la estabilidad de los tiempos de respuesta. La conclusión a la que se puede llegar es que aún no se ha alcanzado el techo de rendimiento del servidor. Para intentar alcanzarlo ejecutaremos una prueba mucho más agresiva, con 40 clientes jugando 20 partidas.

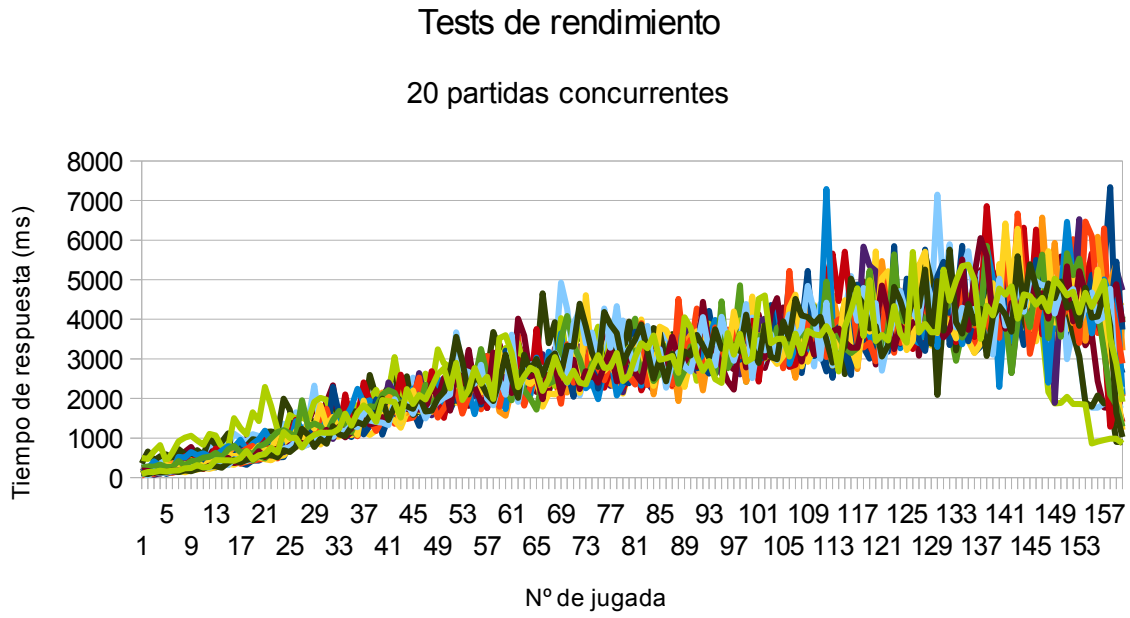

Figura 18: Tiempos de respuesta: 20 partidas concurrentes.

Los tiempos de la ejecución concurrente de 20 partidas, como se muestra en la Figura 18, son considerablemente peores que los obtenidos en las anteriores pruebas. Con 20 partidas de longitud media hemos conseguido sobrepasar la capacidad del servidor, obteniendo tiempos de respuesta hasta 8 veces peores que los de una única partida.

5.1.3 Conclusiones

Tras las pruebas realizadas podemos concluir que el servidor parece preparado para mantener varios cientos de clientes sin excesivos problemas. Aún así, hemos localizado un claro objetivo para las optimizaciones pues, si las partidas de más de 100 movimientos fueran comunes, buena parte del tiempo de procesamiento se dedicaría a ellas.

Es notable que, a pesar de que durante la última prueba el servidor tenía sus cuatro núcleos

trabajando a pleno rendimiento, la carga en RAM se mantuvo casi a niveles de uso cotidiano. Por tanto, en caso de desear instalar el sistema en equipos de gama profesional, convendría centrarse en equipos con gran potencia de cálculo, en lugar de suplir requisitos de memoria inexistentes.

5.2 Usabilidad

En primer lugar analizaremos la usabilidad de la API ofrecida por el servidor, que afecta directamente a la facilidad de creación de clientes, para después exponer una serie de observaciones sobre el cliente desarrollado.

5.2.1 Usabilidad de la API

El hecho de haber desarrollado un cliente como parte del proyecto nos ha permitido comprobar de primera mano las facilidades y limitaciones que el diseño del servidor impone en las aplicaciones que hagan uso de él.

En términos generales hemos podido comprobar que la API responde fácilmente a las necesidades básicas de los clientes, facilitando el acceso a las partidas, invitaciones y peticiones de partidas que los usuarios puedan realizar. Además, lo hace dando libertad a los clientes para decidir cómo realizar los emparejamientos, abriendo la puerta a comportamientos adaptados a los deseos del usuario.

Las únicas limitaciones las hemos encontrado en la petición de partidas recientemente terminadas, para lo que la API no ofrece ningún tipo de soporte. La solución a la que hemos recurrido en el cliente de pruebas es solicitar todas las partidas del jugador, que podrían ser filtradas *a posteriori*, aunque con el correspondiente problema de escalabilidad para jugadores con gran cantidad de partidas terminadas.

Otro problema que hemos podido encontrar es la relativa dificultad para analizar el progreso de las partidas sin recrearlas. Para poder determinar cómo ha terminado una partida o quién ha ganado es necesario evaluar todos los movimientos. Podría resultar conveniente añadir campos con esta información en la base de datos y las clases de intercambio de información, aunque esto implicaría duplicar información y la necesidad de asegurar que se mantiene correctamente actualizada en todos los casos.

5.2.2 Usabilidad del cliente

El principal objetivo que pretendíamos alcanzar con el desarrollo del cliente era evaluar cuán fácil es utilizar la funcionalidad ofrecida por el servidor para ofrecer un entorno de juego completo y simple. Además, era nuestra intención desarrollar un primer prototipo de aplicación, para evaluar diferentes aproximaciones. Las conclusiones referentes al primer punto se han explicado en el apartado anterior, por lo que nos referiremos ahora exclusivamente a la facilidad de uso que el cliente ofrece.

En primer lugar, debemos destacar que la elección de XNA como plataforma gráfica no ha resultado exitosa, pues contábamos con poder encontrar alguna biblioteca potente con funcionalidad como entrada de teclado, componentes gráficos y similares. Como no fue posible hacerse con semejante biblioteca, recurrimos a desarrollar una pequeña colección de componentes, con la funcionalidad básica para nuestros objetivos.

Puesto que los esfuerzos no se han centrado en completar esta implementación ni en hacerla especialmente atractiva, la apariencia del cliente no será la mejor posible. Además, la carencia de un sistema de entrada de texto impide incluir las opciones de crear cuentas o identificarse tecleando un password.

En términos de interacción básica, por otra parte, los usuarios que han probado nuestra aplicación han mostrado una opinión favorable al comportamiento y aspecto de los botones de selección. La organización en submenús también la consideramos satisfactoria, así como la presentación de información sobre partidas e invitaciones en pantalla. A este respecto, la crítica más común se refirió a la incapacidad de invertir el tablero cuando el jugador usa las piezas negras, pues esto confunde a los usuarios.

En lo que a aspectos negativos se refiere, el principal problema lo ha dado el componente de navegación de opciones, usado para las invitaciones y las partidas en curso. La interacción con este componente no resulta clara, por lo que debería buscarse una mejor representación de la existencia de más elementos y cómo acceder a ellos. La opción más lógica, viendo cómo este problema se ha resuelto en otros sistemas, es mostrar vistas parciales de los elementos anterior y posterior a cada lado de la opción actualmente seleccionada.

Capítulo 6: Líneas futuras

En este capítulo presentaremos posibles mejoras que se pueden introducir en el servidor, con vistas a su uso comercial.

6.1 Mejora de seguridad de usuarios

En la versión actual de la aplicación, las claves de los usuarios se están almacenando en texto plano en la base de datos y a la hora de validar una petición de identificación éstas se comparan sin más. Ésta no es una política válida, ni desde el punto de vista legal ni de diseño, y tendrá que ser actualizada en caso de almacenar información de personas físicas.

Para solucionar este problema se almacenará la clave encriptada, tras añadirle una pequeña modificación basada en la cuenta del usuario.

6.2 WSHttpBinding para .NET Compact Framework

La primera mejora a llevar a cabo será crear una interfaz equivalente a la de WSHttpBinding en .NET Framework. Esto permitirá implementar el cliente para teléfonos WP7, haciendo un uso completo del servidor. El principal requisito que debe cumplir es que gestione de forma autónoma las sesiones de los clientes.

6.3 Optimización del motor ChessAsin

En su versión actual, como se puede ver en el apartado de rendimiento, el tiempo necesario para recrear una partida puede resultar excesivo. Convendría explorar posibles optimizaciones para esa parte del código.

6.4 Glicko

Implementar Glicko 2 en el servidor, para sustituir al Elo.

6.5 Mejoras del sistema de clasificación

La puntuación de cada jugador debe ser almacenada al principio de cada partida, para más tarde permitir una correcta conversión a PGN. Además, al actualizar el Elo debe usarse la puntuación que

ambos jugadores tenían al inicio, no una vez la partida ha terminado.

Además, deberían aplicarse tramos al cálculo de la puntuación Elo, con valores de K entre 25 y 10. Para esto se tendrá que guardar la puntuación máxima que cada jugador ha tenido en su historia.

6.6 Variantes y comentarios

Añadir soporte para variantes en las partidas, con las que estudiar juegos alternativos. Además, permitir comentar partidas una vez hayan terminado. Ambos deberían ser almacenados como alternativas, para que cada jugador pueda añadir sus propias variantes y comentarios a partidas y compartirlos con otros.

6.7 PGN

Permitir convertir las partidas a formato PGN, para poder enviarlas por correo o estudiarlas en otro software.

6.8 Funcionalidad de red social

Añadir elementos de red social, como la posibilidad de hablar sobre partidas o crear y gestionar amigos.

6.9 Interfaz de cliente

Completar la funcionalidad del sistema de interfaz de usuario creado, haciéndolo más agradable. Como alternativa, podría crearse un nuevo cliente usando Silverlight, mejor adaptado a la creación de interfaces de usuario que XNA.

6.10 Inteligencia artificial

Podría adaptarse una de las múltiples inteligencias artificiales de ajedrez disponibles en C#, para permitir ejecutar partidas individuales. Debe tenerse en cuenta que la inteligencia artificial elegida debe poder licenciarse bajo términos compatibles con GPLv3. Además, tendrá que adaptarse la representación del tablero a la del motor elegido.

Capítulo 7: Conclusiones

En esta sección analizaremos en qué medida hemos cumplido con las expectativas planteadas al principio del proyecto. También evaluaremos la viabilidad del proyecto como un sistema real a utilizar en servidores públicos.

7.1 Revisión de objetivos

Una vez alcanzado el final del proyecto podemos evaluar en qué grado hemos conseguido cumplir con los objetivos planteados al inicio. A continuación trataremos cada uno de ellos.

7.1.1 Base de datos

Contamos con una base de datos creada en MySQL, con la que podemos almacenar y recuperar toda la información necesaria para el funcionamiento del sistema.

7.1.2 Biblioteca de acceso a la base de datos

El proyecto ChessServiceDatabase nos permite comunicar adecuadamente con la base de datos, facilitando el acceso a la misma desde otros componentes .NET. Esto se hace exclusivamente a través de una conexión local, reduciendo el riesgo de ataques contra la base de datos, pues ésta no se encuentra expuesta a Internet.

Además, se han creado gran cantidad de pruebas unitarias que nos dan confianza en el correcto funcionamiento de esta biblioteca.

7.1.3 Implementación de un motor de ajedrez

ChessAsin es un motor de ajedrez funcional, con soporte para partidas completas cumpliendo con las reglas del deporte. Para probar su fiabilidad se han reproducido centenares de partidas, ejecutándose todas ellas sin problemas.

Los puntos a mejorar que se han encontrado son la velocidad con que se reproducen partidas de duración media y la excesiva complejidad de algunas operaciones, como la coronación de peones. Estos puntos podrán ser solventados en versiones futuras del motor.

7.1.4 Implementación del servidor de juego

El proyecto ha concluido con un servidor operativo, con soporte para registro e identificación de usuarios, baja y eliminación de cuentas, invitaciones y partidas. Con un cliente completo es posible realizar todas las tareas básicas que se podría pedir a un sistema de esta naturaleza.

El servidor cuenta además con funcionalidad no expuesta a los clientes, para facilitar ciertas tareas de administración.

7.1.5 Cliente de juego

El cliente desarrollado para Windows ha demostrado que es posible, con la interfaz expuesta por el servidor, crear una aplicación que permita a varios jugadores competir entre ellos a través de Internet. Aunque el cliente no aprovecha todas las posibilidades que el servidor ofrece, sirve como prueba de concepto para proyectos posteriores.

El principal aspecto a destacar en lo que respecta a este objetivo es que la elección de XNA como plataforma interactiva no fue la más acertada. Silverlight habría sin duda facilitado las cosas, puesto que habría evitado tener que programar los elementos de interfaz. De este modo habríamos contado con más tiempo para el desarrollo de funcionalidad más compleja, como la creación de cuentas desde la interfaz o un sistema de identificación adecuado.

Capítulo 8: Planificación y presupuesto

En este capítulo analizaremos el proceso de planificación del proyecto, comparándolo con el proceso de desarrollo real. Después realizaremos una estimación de los costes en que el proyecto ha incurrido.

8.1 Planificación

Presentamos aquí la división en tareas del proyecto, para más tarde establecer los plazos de desarrollo de cada una.

1. Motor de ajedrez: desarrollo de ChessAsin, el motor de ajedrez que usaremos en el servidor y el cliente.
2. Prototipo: desarrollo de un cliente prototipo en XNA, funcional en Windows Phone 7.
3. Base de datos:
    a. Diseño: determinar las tablas y tipos a utilizar.
    b. Instalación: evaluación, selección e instalación de la base de datos a utilizar.
    c. Implementación: consultas SQL para crear y gestionar la base de datos y sus requisitos.
4. Servidor:
    a. Biblioteca de gestión de la base de datos:
        i. Diseño: estudio del sistema de comunicación con la base de datos y diagrama de clases.
        ii. Implementación: creación de la biblioteca.
        iii. Pruebas unitarias: pruebas completas de la funcionalidad ofrecida, asegurando la consistencia de la información almacenada y obtenida.
    b. Diseño del servidor: estudio de las características de WCF y diseño en consecuencia del servidor. Primera estimación de los parámetros de configuración del servidor.
    c. Implementación del servidor: implementar el servidor como se ha planificado y probarlo a *grosso modo*.
    d. Pruebas unitarias del servidor: creación de pruebas, mediante ejecuciones de partidas

completas y comportamientos especiales.

e. Instalación y configuración del servidor: configurar el servidor, según los parámetros estimados anteriormente, y modificarlos hasta conseguir conectar remotamente al servidor.

5. Desarrollo del cliente:

   a. Biblioteca gráfica: implementar una biblioteca de controles de interfaz adaptada a nuestras necesidades.

   b. Implementación: crear un cliente completo capaz de conectar con el servidor, gestionar partidas en curso y actualizar la información del servidor según los deseos del usuario. Esta fase no requiere de diseño por haberse desarrollado en ciclos de prototipos.

   c. Configuración de red: configuración de los parámetros de red para poder conectar con un servidor remoto, en lugar del entorno de pruebas controlado.

6. Documento de proyecto: aunque cada fase anterior incluye la documentación pertinente del código o los procesos involucrados, es necesario dedicar tiempo a la redacción del documento global del proyecto.

8.1.1 Planificación original

A continuación presentamos la planificación original para el proyecto, usando un diagrama de Gantt. Debe tenerse en cuenta, a la hora de evaluar esta planificación, el hecho de que la mayor parte del proyecto se desarrolló a la par que otras ocupaciones laborales y proyectos, por lo que se estimó una jornada a tiempo parcial de unas 6 horas diarias.

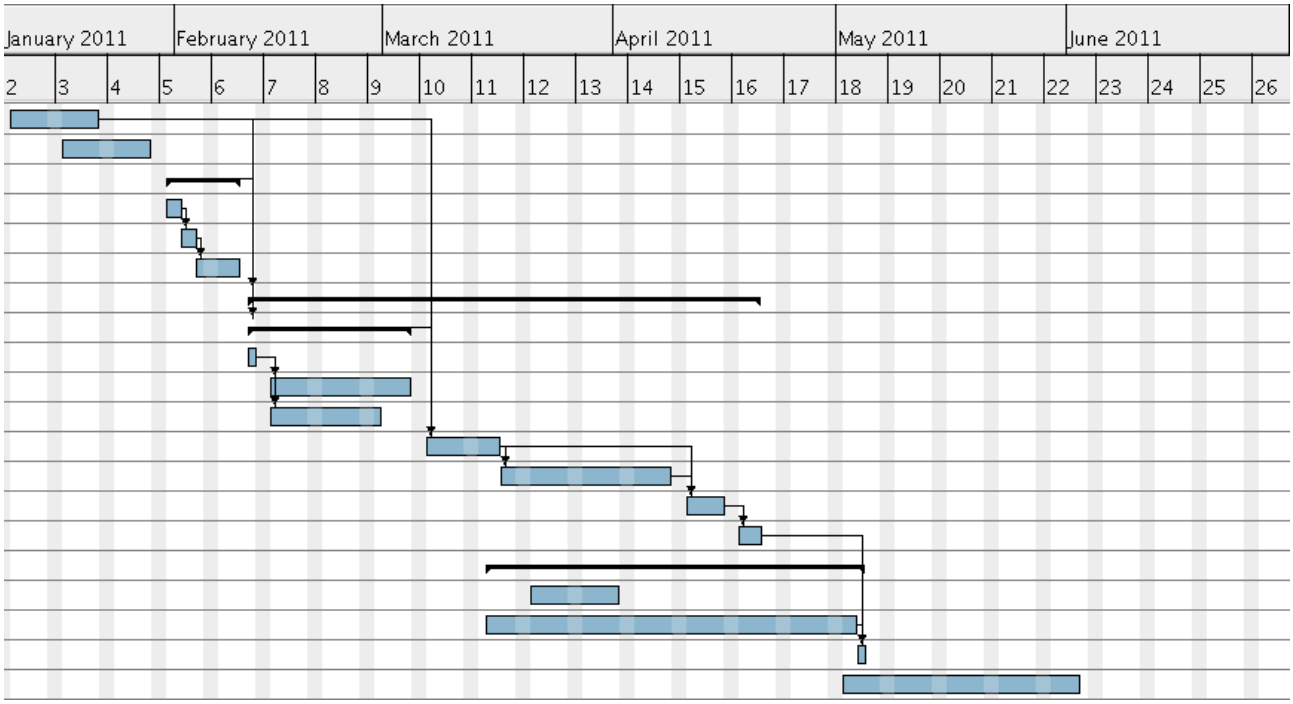

Figura 19: Diagrama de planificación original.

8.1.2 Evolución real del desarrollo

Debido a la cantidad de tecnologías poco conocidas que se han utilizado en el proyecto, ninguna de las fases anteriormente descrita fue llevada a cabo de forma aislada. En el resultado final, se dieron gran cantidad de solapamientos, pausas y cambios en cómo se desarrolló el servidor. Sirva como ejemplo el hecho de que en fases avanzadas del desarrollo aún se descubrieron fallos menores en el motor de juego. El tiempo que se dedicó a estas tareas se contabilizará como parte del proceso en que tuvo lugar.

Durante el desarrollo surgieron además problemas no previstos, normalmente relacionados con la instalación o configuración de las tecnologías a usar. En particular, la elección de base de datos, originalmente Microsoft SQL Server, se alargó durante mucho más tiempo del previsto, por problemas a la hora de instalar el software y conseguir comunicarse con él desde un proyecto de Visual Studio. Tras varios días de pruebas se decidió probar suerte con MySQL, que se instaló y configuró satisfactoriamente en un día.

Un problema similar surgió con la configuración de WCF, que se alargó durante todo el proyecto, hasta que finalmente se consiguió tener una instalación completamente funcional. La escasa

documentación y gran cantidad de casos particulares en las opciones de configuración hicieron especialmente complicada esta fase del proyecto.

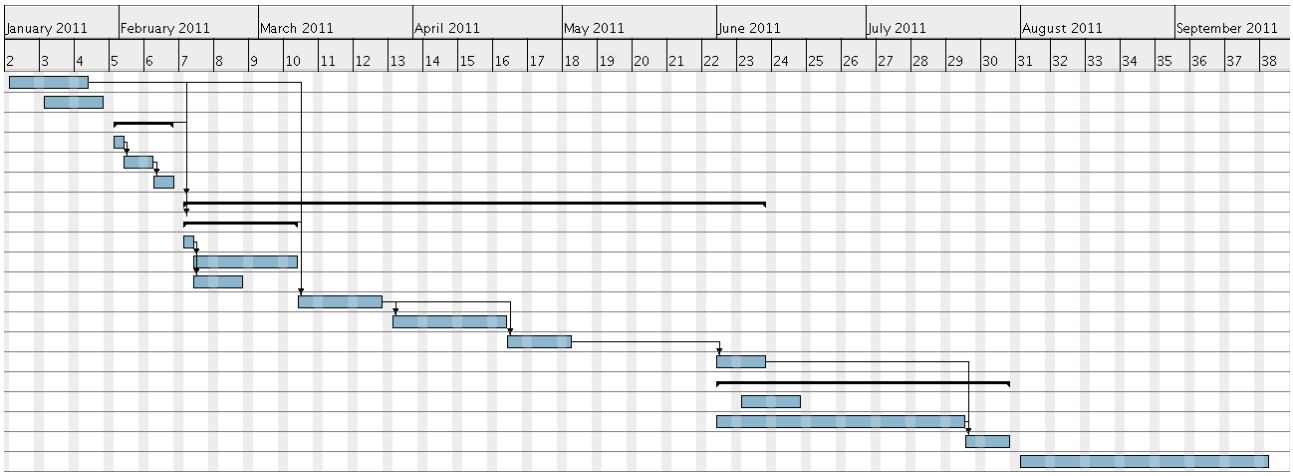

Figura 20: Diagrama de desarrollo real.

Como se puede apreciar en el diagrama, el desarrollo se detuvo durante el mes de Mayo, por causas ajenas al proyecto.

8.2 Recursos

A continuación detallaremos el equipo hardware y herramientas software utilizadas durante el desarrollo del proyecto. También se indicará el precio de cada elemento, para más tarde poder evaluar los costes en que se incurrió durante el proyecto.

Debido al amplio uso de aplicaciones *Open Source* o gratuitas, buena parte de los costes se mantienen en 0€. Además, el precio del Windows 7 Professional usado en el cliente viene incluido en el coste del mismo, por ser un equipo portátil comprado junto con la licencia del sistema operativo.

Sección	**Equipamiento**	**Precio**
Hardware	Servidor / equipo de desarrollo	800 €
	Cliente / equipo de desarrollo	700 €
Software	Windows 7 Professional (servidor)	250 €[17]
	Visual Studio Express (x2)	0 €

17 http://www.microsoft.com/uk/windows/buy/default.aspx

Sección	Equipamiento	Precio
	MySQL	0 €
	MySQL Workbench	0 €
	Microsoft IIS	0 €
	TortoiseSVN	0 €
	Doxygen	0 €
	Dia	0 €
	GanttProject	0 €
	GIMP	0 €
	LibreOffice Writer	0 €
Servicios	Assembla SVN	0 €
	Sourceforge project hosting	0 €
Recursos humanos	Programador / analista	24.000 € / año[18]

8.3 Análisis económico

Para la evaluación de los costes finales se tendrán en cuenta los costes proporcionalmente al tiempo que fueron utilizados durante el proyecto, respecto a la vida útil estimada de cada uno de ellos. Esto se debe a que, una vez terminado el mismo, los equipos y programas adquiridos podrán seguir siendo utilizados en futuros desarrollos.

Sólo se detallarán los costes de aquellos componentes con precio mayor de 0€.

8.3.1 Costes estimados

En primer lugar detallaremos los costes estimados, de haberse cumplido los plazos originalmente planificados.

18 Con jornada a tiempo parcial de 6 horas al día.

Recurso	Precio	Vida útil estimada	Uso estimado	Coste para el proyecto
Servidor	800€	72 meses	5 meses	55,55 €
Cliente	700€	48 meses	5 meses	72,91 €
Windows 7 Professional	250 €	120 meses	5 meses	10,41 €
Programador / analista	24.000 €	12 meses	5 meses	10.000 €
Total				**10.138,87 €**

Tabla 19: Costes estimados del proyecto.

8.3.2 Costes finales

Debido al retraso en el desarrollo, los costes reales superan los estimados. A continuación se presenta el coste final del proyecto.

Recurso	Precio	Vida útil estimada	Uso real	Coste para el proyecto
Servidor	800€	72 meses	8 meses	88,88 €
Cliente	700€	48 meses	8 meses	116,66 €
Windows 7 Professional	250 €	120 meses	8 meses	16,66 €
Programador / analista	24.000 €	12 meses	8 meses	16.000 €
Total				**16.222,2 €**

Tabla 20: Costes finales del proyecto.

Apéndice A: Instalación de la base de datos

La base de datos que hemos usado para el proyecto es MySQL, en sus versiones 5.1 y 5.5. Puesto que usamos algunas características introducidas en MySQL 5.1, no se recomienda el uso de versiones anteriores. Además, hemos utilizado MySQL Workbench v.5.2.34.2, aunque no es necesario para la gestión de la base de datos.

Aunque no es obligatorio, recomendamos seleccionar *InnoDB* como el formato de tablas por omisión, y *UTF-8* como el formato de texto preferido.

Para la creación de las tablas y el usuario requerido se han creado una colección de scripts de SQL en la carpeta *MySQL\ChessDB*.

- CreateSchema.sql: Crea el esquema, *chessdb*, y las tablas y vistas para acceder a los datos.
- CreateUser.sql: Crea el usuario *chessClient* y le otorga los permisos que necesita para gestionar el esquema *chessdb*. Éste es el usuario que la biblioteca de acceso a la base de datos utiliza. A *chessClient* se le permitirá acceder a la base de datos sólo de forma local.
- InsertTest.sql: Añade datos a la base de datos, para permitir hacer pruebas.

Apéndice B: Instalación del servidor

El servidor usado en el proyecto ha sido montado sobre Microsoft Internet Information Services (IIS7), instalado desde Web Platform Installer 3.10 (WPI).

Los pasos necesarios para configurar el servidor, una vez instalado, son:

Habilitar el bridge de ASP.NET para IIS, ejecutando desde una consola con privilegios de administrador[19]:

c:\Windows\Microsoft.NET\Framework64\v4.0.30319\aspnet_regiis.exe -i

Habilitar la gestión de archivos svc por ASP:

"c:\Windows\Microsoft.NET\Framework64\v3.0\Windows Communication Foundation\ServiceModelReg.exe" -i

Ejecutar el gestor de IIS, *inetmgr*, como administrador y configurarlo como sigue:

Sites -> Add Web Site...

Name: ChessAsin

Port: 15432

Physical path: ruta a una copia local del directorio IISWebSite del repositorio. Esta copia local debe encontrarse en un sistema de archivos NTFS para evitar problemas.

Connect as: [cuenta y clave de la cuenta de administrador del sistema]

Protocol: http

Start immediately: True

Test settings...

ChessAsin -> Add Application...

Alias: ChessAsin

Application pool: ChessAsin

19 En los comandos de consola, adaptar las rutas para que se correspondan con las versiones del .NET Framework instaladas y con la longitud de palabra del sistema operativo (32 ó 64 bits).

Physical path: El mismo que en Site.

Connect as: El mismo que en Site.

Test settings...

ChessAsin -> Add virtual directory

Alias: IISChessService

Physical path: el mismo que en Site.

Connect as: el mismo que en Site.

Test settings...

Seleccionar la web ChessAsin y entrar en la configuración de Modules. Eliminar la entrada de ServiceModel con versión 3.*, para sólo dejar la de versión 4. Esto hará que la web utilice siempre una versión de .NET compatible con la usada en el proyecto.

Application Pools, ChessAsin -> Advanced settings...

Establecer la versión de .NET Framework a 4.0, una vez más para que coincida con la usada en el proyecto.

Crear un directorio *bin* dentro del directorio donde se aloja la web (*IISWebSite*). Copiar los binarios compilados de ChessServiceLibrary en esa carpeta. Todas las dlls (ChessServiceLibrary, ServerChessDatabase y WinChessAsin), más ChessServiceLibrary.dll.config son necesarios. Los archivos de símbolos (pdb) son opcionales.

Apéndice C: Instrucciones de uso del cliente

Puesto que es una prueba de concepto, los usuarios vienen ya preparados para las cuentas de prueba del sistema. La pantalla de selección de usuario es una pantalla más del sistema, por lo que es posible volver a ella más adelante.

Para navegar entre los diferentes menús, seleccione las opciones deseadas con el ratón. Los botones seleccionados aumentarán de tamaño y cambiarán de color para resaltar. Si desea volver a la pantalla anterior, utilice la tecla *Esc*, usada como sustituto de la tecla de retroceso de los teléfonos Windows Phone 7.

En los menús de navegación de opciones (invitaciones y partidas), puede deslizar el ratón por encima de la opción actual (parte izquierda de la pantalla) de izquierda a derecha o viceversa, mientras mantiene el botón izquierdo del ratón apretado. Pequeñas flechas en la parte superior izquierda o inferior derecha le indicarán si existen elementos adicionales a la izquierda o derecha, respectivamente. Si no ve dichas flechas, el elemento actual es el primero o el último de la lista.

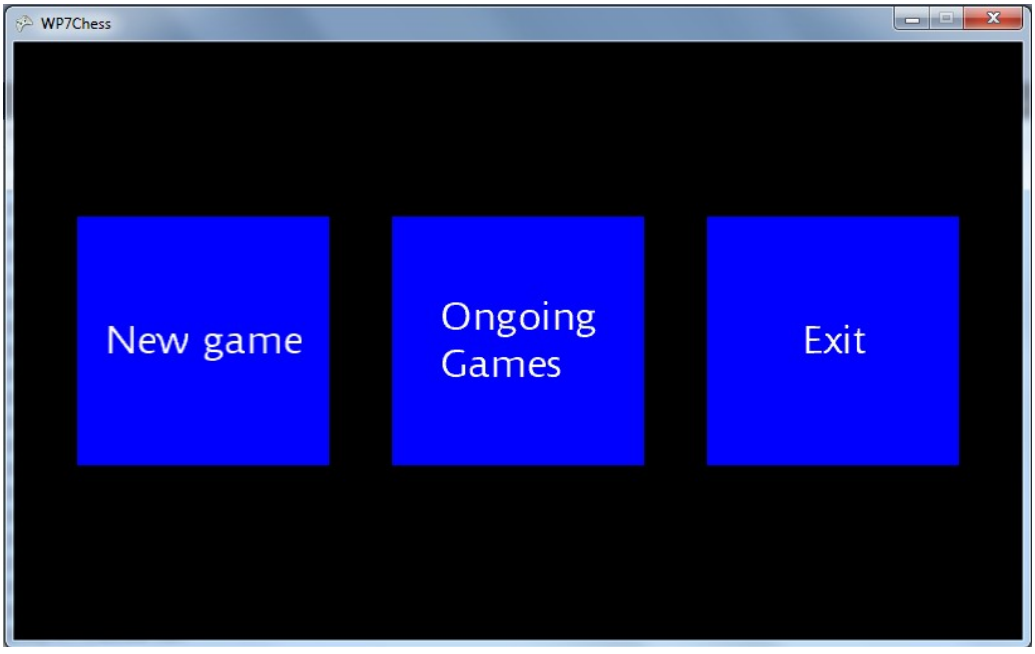

Figura 21: Pantalla inicial.

Desde la pantalla inicial puede elegir iniciar una nueva partida o continuar una ya en curso. Las opciones para nueva partida son:

- Partida de práctica: jugada localmente, con el usuario moviendo para cada uno de los dos

bandos. Abandonar el juego reseteará la partida, si más adelante vuelve a elegir esa opción.

- Crear invitación: invitar a jugar a otro jugador. En el cliente de prueba sólo se puede retar al otro usuario habilitado. Puede elegir el color con que desea jugar en una segunda ventana.
- Ver invitaciones: navegar por la lista de invitaciones recibidas, observando la puntuación Elo del rival. Las invitaciones pueden ser aceptadas o rechazadas. Si se acepta una invitación, deberá ir al menú de partidas en curso para jugarla.
- Solicitar rival: se ofrecerá la opción de limitar el rango Elo del rival, comparado con el del usuario, siendo las opciones menor que, menor o similar, similar, mayor o similar, mayor o indiferente. Si otro jugador ha solicitado un rival y tanto el usuario como el potencial rival cumplen los requisitos exigidos por el otro, una nueva partida se creará automáticamente. En caso contrario, la petición se guardará en el servidor a la espera de que alguien realice una petición compatible.

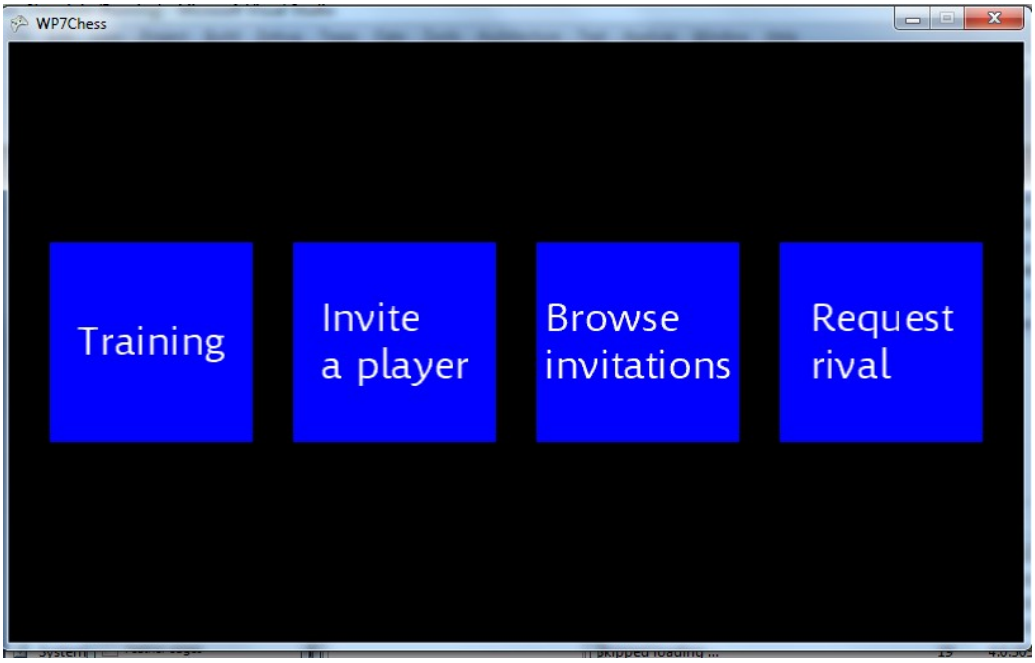

Figura 22: Menú de nueva partida.

Una vez se tienen juegos en curso se puede acceder a la lista de partidas. En esta se puede observar el progreso de cada partida jugada por el usuario. Si el usuario selecciona el tablero con el botón izquierdo del ratón pasará a jugar la partida. Si le corresponde el turno, podrá realizar un movimiento. Para salir de la partida se puede pulsar *Esc*.

El movimiento realizado no se actualizará en el servidor automáticamente. Para ello el usuario debe

confirmarlo seleccionando el botón de confirmación. Sin embargo, si no le gusta el movimiento, puede decidir cancelarlo con el botón *X*. El botón de refresco actualizará la lista de partidas con aquéllas en las que los rivales hayan realizado un movimiento o las partidas que se hayan iniciado recientemente. Los movimientos realizados por el jugador que no hayan sido confirmados no se perderán al usar esta opción.

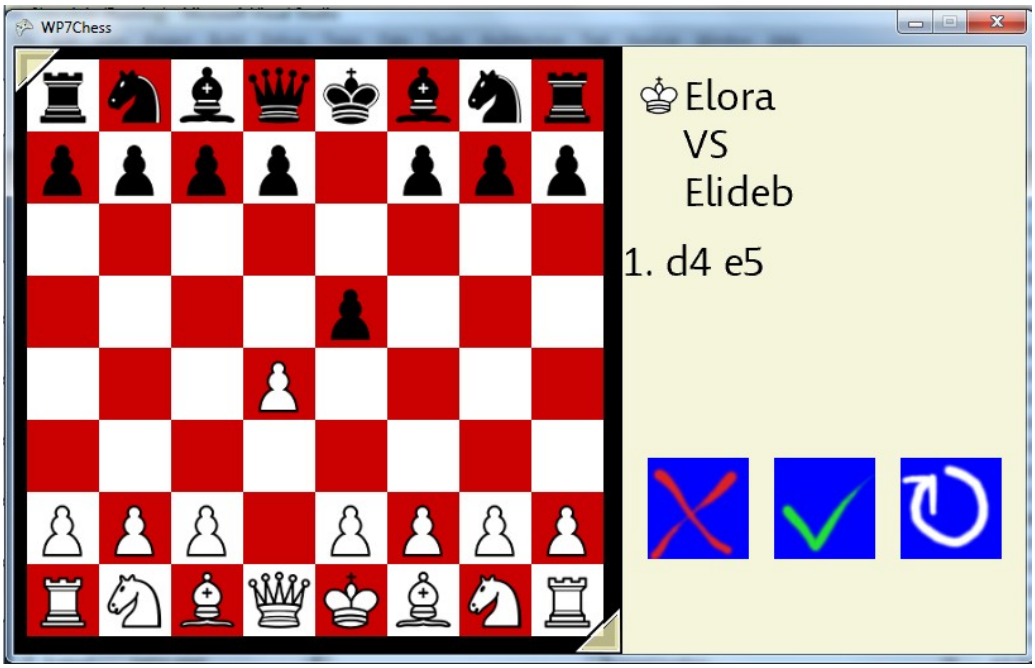

Figura 23: Menú de partidas disponibles.

Dentro de una partida existen varias opciones de interacción. Los movimientos de piezas se realizan pulsando sobre la pieza a mover y arrastrándola a la posición de destino, mientras se mantiene el botón izquierdo del ratón pulsado. Al soltar el botón se ejecutará el movimiento, si éste es válido.
Para ejecutar enroques se debe mover el rey. Si el rey se mueve dos posiciones en horizontal y el enroque es posible, éste se ejecutará automáticamente.
En caso de coronar un peón se presentará la opción de seleccionar la pieza deseada en el menú lateral izquierdo. Tras realizar la selección se ejecutará el movimiento.
Para ver qué jugadas han llevado a cabo los jugadores se puede arrastrar el listado de movimientos hacia arriba o abajo, revelando jugadas anteriores o posteriores.

Figura 24: Partida en curso.

También existen las opciones de rendirse o solicitar tablas, pero se encuentran ocultas para evitar el uso accidental. Para ver estas opciones debe arrastrarse la barra que separa el tablero de la lista de movimientos hacia la izquierda, hasta que los botones se revelen. Tras seleccionar la opción deseada, o si cambia de opinión, puede volver a esconder los botones arrastrando de nuevo la barra de separación a la derecha.

La rendición y declaración de tablas también deben ser confirmadas para que se comuniquen al servidor.

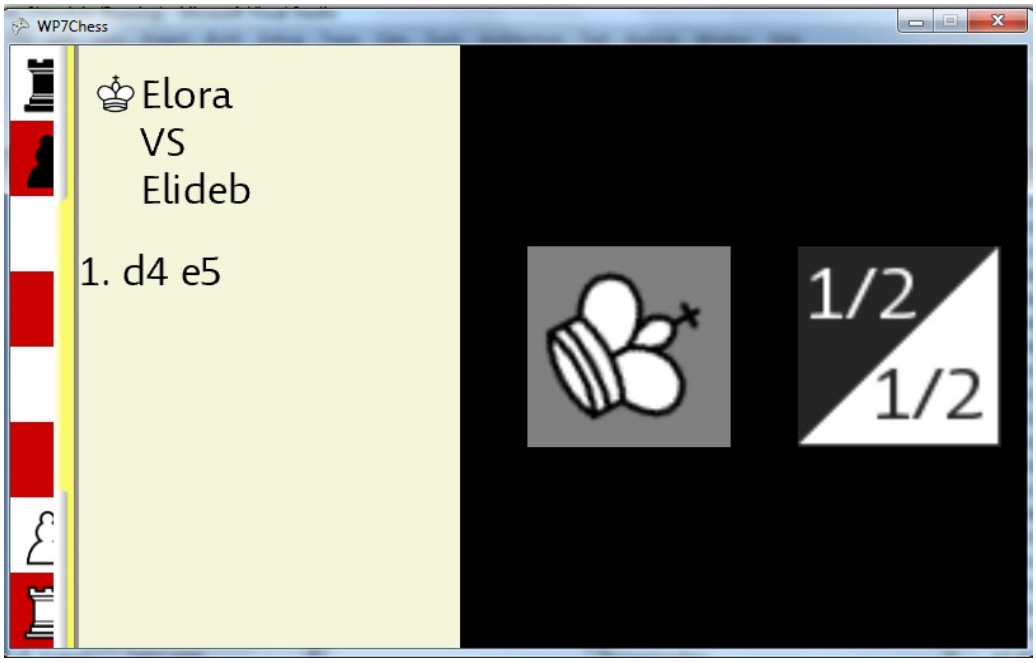

Figura 25: Botones de rendición y tablas.

Tenga en cuenta que, en la versión actual del cliente, la información del cliente no se mantiene tras cerrar la aplicación. Por tanto, todo movimiento no confirmado se perderá si se apaga el sistema.

Bibliografía

[1] Encyclopedia Britannica, “chess (game).” [Online]. Available: http://www.britannica.com/EBchecked/topic/109655/chess. [Accessed: 06-Nov-2011].
[2] FIDE, “FIDE Handbook: Laws of Chess,” Nov-2008. [Online]. Available: http://www.fide.com/fide/handbook.html?id=124&view=article. [Accessed: 06-Nov-2011].
[3] S. Johnson, “Analysis: Asynchronicity In Game Design,” *Gamasutra*, 07-Jul-2009. [Online]. Available: http://gamasutra.com/view/news/24310/Analysis_Asynchronicity_In_Game_Design.php. [Accessed: 06-Nov-2011].
[4] M. E. Glickman, “The Glicko system.” Boston University.
[5] M. E. Glickman, “Example of the Glicko-2 system.” Boston University.
[6] FIDE, “FIDE Handbook: Appendices.” [Online]. Available: http://www.fide.com/fide/handbook.html?id=125&view=article. [Accessed: 06-Nov-2011].
[7] A. Orlowski, “Symbian’s Secret History: The battle for the company’s soul • The Register,” 29-Nov-2010. [Online]. Available: http://www.theregister.co.uk/2010/11/29/symbian_history_part_two_ui_wars/. [Accessed: 07-Nov-2011].
[8] DISTIMO, “Distimo Publication - April 2011.” Apr-2011.
[9] DISTIMO, “Distimo Publication - May 2011.” May-2011.